

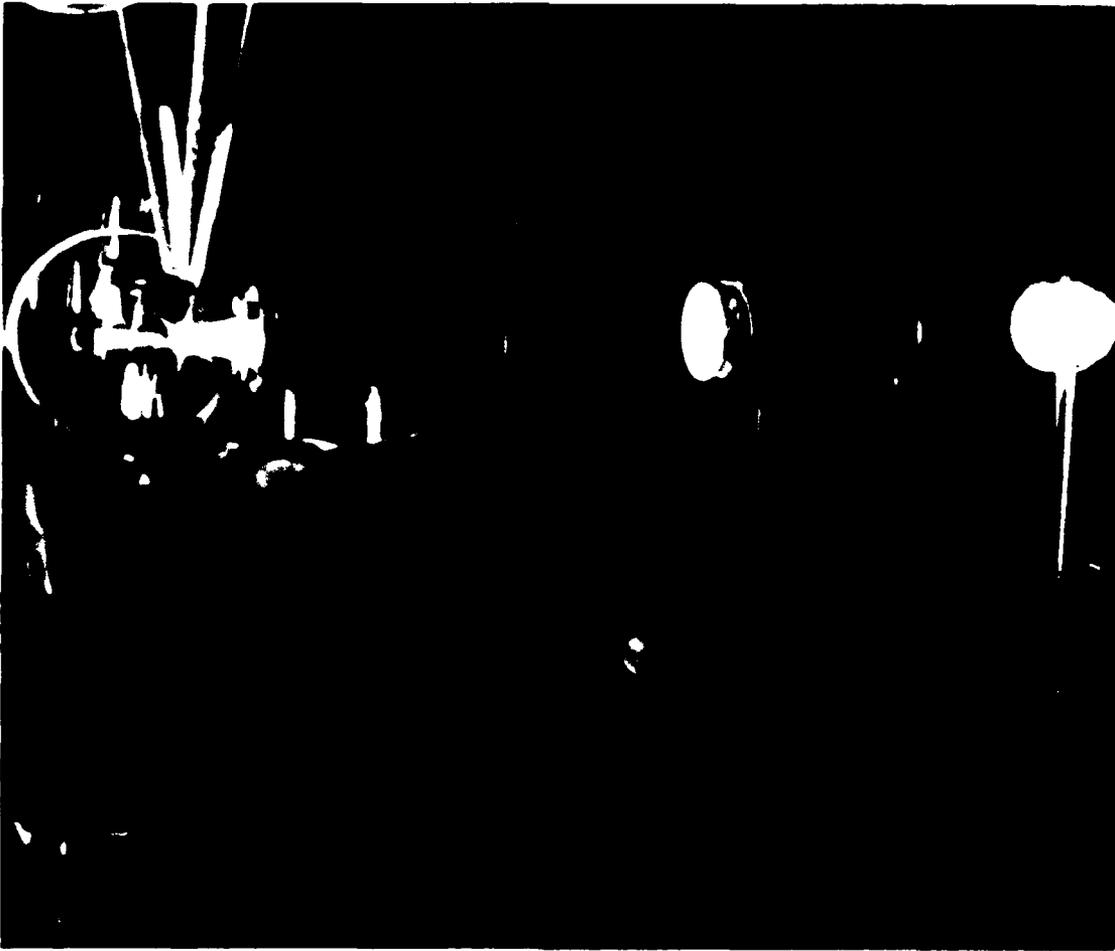
**.kms
fusion**
inc

AD-A150 513

Parallel Processing for Computational Continuum Dynamics*

Joseph F. McGrath
Darrell L. Hicks
Lorie M. Liebrock

**DTIC
SELECTED**
FEB 19 1985
S E



DTIC FILE COPY

Reprint
January, 1985

submitted
Workshop on Parallel Processing
Using the Heterogeneous Element Processor
University of Oklahoma
March 20 and 21, 1985

Approved for public release
Distribution unlimited.

02 06 038

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				
1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b RESTRICTIVE MARKINGS		
2a SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.		
2b DECLASSIFICATION/DOWNGRADING SCHEDULE				
4 PERFORMING ORGANIZATION REPORT NUMBER(S) KMSF-U1539		5 MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 85 - 0045		
6a NAME OF PERFORMING ORGANIZATION KMS Fusion, Inc.	6b OFFICE SYMBOL <i>(If applicable)</i>	7a NAME OF MONITORING ORGANIZATION Air Force Office of Scientific Research		
6c ADDRESS (City, State and ZIP Code) 3621 South State Road P.O. Box 1667 Ann Arbor MI 48106		7b ADDRESS (City, State and ZIP Code) Directorate of Mathematical & Information Sciences, Bolling AFB DC 20332-6448		
8a NAME OF FUNDING/SPONSORING ORGANIZATION AFOSR	8b OFFICE SYMBOL <i>(If applicable)</i> NM	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F49620-84-C-0111		
8c ADDRESS (City, State and ZIP Code) Bolling AFB DC 20332-6448		10 SOURCE OF FUNDING NOS		
		PROGRAM ELEMENT NO 61102F	PROJECT NO 3005	TASK NO A1
		WORK UNIT NO		
11 TITLE (Include Security Classification) PARALLEL PROCESSING FOR COMPUTATIONAL CONTINUUM DYNAMICS				
12 PERSONAL AUTHOR(S) Joseph F. McGrath, Darrell L. Hicks* and Lorie M. Liebrock*				
13a TYPE OF REPORT Reprint	13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Yr., Mo., Day) January 1985		15 PAGE COUNT 19
16 SUPPLEMENTARY NOTATION KMS Fusion, Inc., KMSF-U1539, pp. 1-19, January 1985. Submitted to Workshop on Parallel Processing Using the Heterogeneous Element Processor, Univ of Oklahoma, 20-21 March 1985.				
17 COBAY CODES		18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB GR		
19 ABSTRACT (Continue on reverse if necessary and identify by block number) The numerical solution of many problems in continuum dynamics is seriously limited by the computation rates attainable on computers with serial architecture. Parallel processing machines can achieve much higher rates. However, applying additional processors to a calculation is only part of the solution. In this report, parallel algorithms are developed for explicit and implicit, Lagrangian and Eulerian finite difference schemes for computational continuum dynamics in one spatial dimension. First, the explicit conservation equations in the Lagrangian reference frame are readily reformulated for concurrent processing. Second, an implicit solution is derived for these equations. This is important because it yields unconditional stability. The parallelism is achieved via a block implicit numerical scheme. Third, a rezoning algorithm is employed with each Lagrangian integration step to transform the mesh back to the Eulerian reference frame. Along the algorithmic development path, a zone-by-zone (CONTINUED)				
20 DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS <input type="checkbox"/>		21 ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a NAME OF RESPONSIBLE INDIVIDUAL CPT John P. Thomas, Jr.		22b TELEPHONE NUMBER <i>(Include Area Code)</i> (202) 767- 5026	22c OFFICE SYMBOL NM	

DD FORM 1473, 83 APR

EDITION OF 1 JAN 75 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

ITEM #12, PERSONAL AUTHOR(S): *Michigan Technological University, Houghton MI 49931.

ITEM #19, ABSTRACT, CONTINUED: parallelization gives way to a block-by-block technique both of which are self-scheduling. Then the latter is compared to an approach that keeps the parallel processes alive for many time steps. At each step of this research project, the derived numerical methods provide effective algorithms for exploiting the architectural advantages of the HEP H1000 (Heterogeneous Element processor) computer.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

Parallel Processing for Computational Continuum Dynamics*

Joseph F. McGrath

KMS Fusion, Inc., Ann Arbor, Michigan 48106

Darrell L. Hicks and Lorie M. Liebrock

Michigan Technological University, Houghton, Michigan 49931

March, 1985

Abstract

The numerical solution of many problems in continuum dynamics is seriously limited by the computation rates attainable on computers with serial architecture. Parallel processing machines can achieve much higher rates. However, applying additional processors to a calculation is only part of the solution. In this report, parallel algorithms are developed for explicit and implicit, Lagrangian and Eulerian finite difference schemes for computational continuum dynamics in one spatial dimension.

First, the explicit conservation equations in the Lagrangian reference frame are readily reformulated for concurrent processing. Second, an implicit solution is derived for these equations. This is important because it yields unconditional stability. The parallelism is achieved via a block implicit numerical scheme. Third, a rezoning algorithm is employed with each Lagrangian integration step to transform the mesh back to the Eulerian reference frame. Along the algorithmic development path, a zone-by-zone parallelization gives way to a block-by-block technique both of which are self-scheduling. Then the latter is compared to an approach that keeps the parallel processes alive for many time steps. At each step of this research project, the derived numerical methods provide effective algorithms for exploiting the architectural advantages of the HEP H1000 (Heterogeneous Element processor) computer. ↙

*Research sponsored by the Air Force Office of Scientific Research (AFSC) under Contract F49620-84-C-0111. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Section I - Introduction

The conservation laws of volume, mass, momentum, and energy apply to any continuum material: solid, liquid, gas, plasma, or multiphase. Hence, the algorithms of computational continuum dynamics are very important for the solution of many scientific problems. When the application is changed from one material to another, only the material law (equation of state, constitutive relation, or rate relation) changes. Thus, there is a similarity of structure between the hydrocodes (gas and liquid dynamics), the wavecodes (solid dynamics), and the magnetohydrocodes (plasma dynamics) that are the computer implementations of schemes for continuum dynamics calculations.

It is well known that computer simulation codes are cost-effective tools in continuum dynamics research. Indeed, a variety of problems arising in such fields as aeronautics, controlled fusion, meteorology, reactor safety, and structural analysis provide strong motivation for the development of higher computing rates. However, the limits of current computing power prevent the simulation of many important problems to the desired levels of temporal and spatial resolution. The speed of light barrier imposes a theoretical limit on what can be achieved with serial architecture.

To achieve higher computing rates it has become necessary to perform calculations in parallel. The computer architecture with the greatest degree of parallelism is labeled Multiple Instruction stream, Multiple Data stream (MIMD). An example of a machine of this type is the HEP H1000 computer manufactured by the Denelcor Corporation.

In principle, many hydrocodes and wavecodes could be moved to a parallel processor. However, applying additional processors to a computational task is not, in general, sufficient to produce significant speed-up. Indeed, the development of parallel algorithms is an area of research vital to the effectiveness of parallel processors.¹ Recent research indicates that the parallelization of a program should be organized from the top down.^{2,3} That is, the existing structure and organization of a program do not permit significant improvements in speed. Consequently, it becomes necessary to reformulate algorithms and to write new code.

The direct approach to the construction of parallel algorithms for continuum dynamics calculations can be quite complicated. Rather than plunge

into a development project intended to generate a code with broad three-dimensional capabilities, we have taken a step by step approach suggested by Darrell Hicks.⁴ Thus, in a modular fashion, the algorithms at each level of complexity can be verified as they are derived. Our approach to the numerical solution of the problems of continuum dynamics leads to algorithms well suited for parallel architecture in general and for the HEP H1000 computer in particular. The approach is a step-by-step procedure based on a progression from the simplest hydrocode (one-dimensional, single-phase, explicit, Lagrangian) through the most complex continuum dynamics codes. The latter programs can involve two or three dimensions, multiphase, implicit-optional-explicit differencing, arbitrary rezoning coordinate systems (Lagrangian or Eulerian), or variable time steps from spatial zone to zone.

The specific objectives of the research reported in this paper involve a three-step process for the development of parallel algorithms for one-dimensional simulation codes.⁵ First, we will consider an explicit, one-dimensional, single-phase Lagrangian hydrocode. Its inherently simple data structure makes it straightforward to integrate the volume, momentum, and energy equations for each zone, or block of zones, in parallel. Thus, for this case, the optimal parallelization problem is evidently easily solved. We show that, in some sense, this is the best restructuring of the algorithm. Second, a block implicit method is derived for the implicit differencing of the equations. Third, we will convert the programs from Lagrangian to Eulerian coordinates in such a way that the conserved quantities are preserved in a parallel processing scheme.

Extensions to Eulerian codes can be achieved by a rezoning technique. It has precursors in the work of F. Harlow⁶ (Particle In Cell codes) and in the work of W. Johnson (PIC codes converted to continuum simulations). We refer to this method as the Harlow-Johnson rezoning technique.⁵ It can be modified to achieve dynamic rezoning (also known as adaptive mesh) methods. (See Hicks⁴ and Hicks and Walsh⁷ for further details.)

The results of this work provide the foundation for extensions to more elaborate hydrocodes. A direct extension appears to be possible for two-phase or two-material flow in Eulerian coordinates.⁴ Through stages of increasing complexity, one can achieve the block-by-block parallelization of multiphase (or multi-material) and multi-dimensional hydrocodes. It appears that

block-by-block parallelization or some variation thereof will lead to the optimal parallel algorithms. Two-phase flow extensions are readily accomplished in an Eulerian coordinate system. Discussions of two-phase flow models and their relevance to reactor safety may be found in Hicks^{4,8,9,10,11} and Ransom and Hicks¹². One of the important problems in reactor safety is the need for fast simulators and predictors to assist operators in handling situations such as the event at Three Mile Island. Finally, extensions to multi-dimensions may be achieved by operator splitting.¹³

While the development of hydrocodes is a long standing achievement, it is the utilization of new and unique advances in computer architecture for hydrocode calculations that makes our research important and timely. In each of the three steps, the crucial questions concern the extent to which the algorithms can be separated into calculations that are performed concurrently. The algorithmic details are developed in Section III. Converting to Eulerian coordinates is the topic of Section IV. The computational results are presented in Section V.

The design of a parallel algorithm is interrelated with the particular architecture of the parallel processor.^{1,2,3} That is, although some progress has been made,¹⁴ general specifications for machine-independent algorithms have not yet been agreed upon. In Section II, we begin by providing a description of the architectural implications for parallel algorithms on the HEP computer.

Section II - The HEP Computer

If we categorize computer architectures by their parallel processing capabilities, we have: SISD, SIMD, and MIMD. SISD stands for Single Instruction stream, Single Data stream. The typical serial computer has SISD architecture. SIMD stands for Single Instruction stream, Multiple Data stream. Vector machines such as the CRAY-1 have this architecture. The multiple data streams consist of the components of the vectors. The instruction mode is still single stream (or serial) although the instructions may generate vector operations. MIMD stands for Multiple Instruction stream, Multiple Data stream. The HEP (Heterogeneous Element Processor) by Denelcor has MIMD architecture.

The HEP computer is designed to combine from one up to 16 Process Execution Modules (PEM's) in a single computer. Each PEM is an eight-segment pipelined processor consisting of eight Function Units in the Instruction Processing Unit. The Function Units include the hardware for initiating

parallel processing and for an integer addition, a floating point addition, a multiplication, and several divides. The machine has a 64-bit word length. More information on the HEP architecture and on its applications can be found in the references by Smith² and Jordan.³

HEP Software

From the software point of view, the HEP achieves its parallel processing capabilities by extending FORTRAN 77 in two ways. With their first implementation of a compiler, Denelcor provided the CREATE statement and the asynchronous variable. In their more recent release, they have replaced these with callable subroutines which make the FORTRAN portable. Even though this is a good objective, the authors deplore the change of heart at Denelcor. The CREATE statement and the asynchronous variable gave the programmer more explicit language for coding calculations that are intended to be performed concurrently. In any event, a discussion of these two terms is appropriate for this report because it provides a good explanation of the considerations that must be taken into account when processing is divided into and reunited from parallel paths.

1. The CREATE SUBROUTINE statement is similar (syntactically) to the well-known CALL SUBROUTINE statement in FORTRAN. It has the effect of creating a copy of (or "cloning") the original subroutine and executing the copy in a calculational stream parallel to the mainstream.
2. The asynchronous ("dollar-sign") variable is any acceptable FORTRAN variable name prefixed with a "\$". Asynchronous variables are used for communication between parallel computational streams. Asynchronous variables have two states, "full" and "empty". If a FORTRAN assignment statement contains an asynchronous variable on the right hand side of the equal sign, then the calculation waits until the state of the asynchronous variable is full. If its state is full, then the value is fetched and the state is set to empty. If the left hand side of the equal side of a FORTRAN assignment statement is an asynchronous variable, then the assignment of its value waits until its state is empty. Then, when the assignment is made, its state is reset to full.

The diagram in Figure 1 presents a visual image of the parallel processing as implemented on the HEP computer. At CREATE statements, the computational flow can "fork" into a number of parallel paths which the operating system assigns to the available processors. An empty asynchronous variable prevents the main stream from continuing beyond the point at which it needs the results from the parallel streams. A "join" point is marked by the return from a CREATE statement and an asynchronous variable that must be reset to full by the last parallel stream to finish processing.

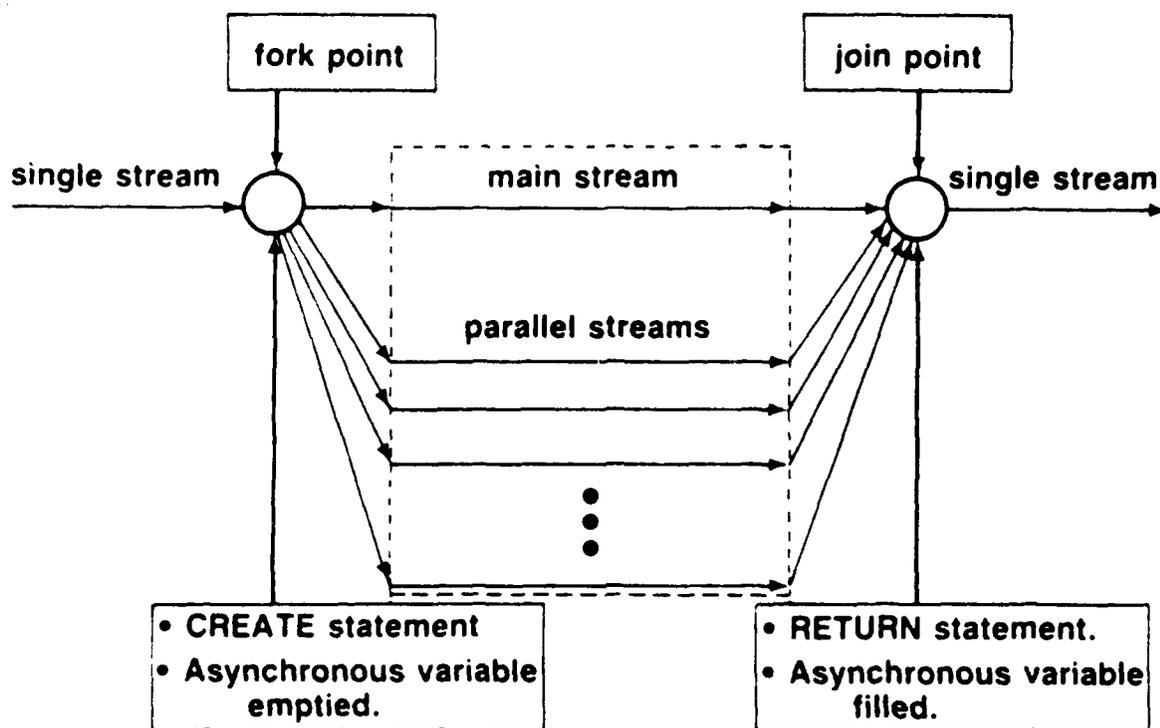


Figure 1. Fork and join points in a FORTRAN program mark the beginning and end, respectively, of concurrent (or parallel) processing segments of the code.

One of the crucial aspects of parallel processing is, of course, the development of software capable of coordinating concurrent computational tasks. Denelcor has chosen a straightforward extension of FORTRAN for the HEP. The "fork" and "join" procedures make the HEP computer immediately accessible to the traditional scientific programming community. A collection of lectures on various aspects of concurrent computation contains further background material on parallel processing algorithms and architecture.¹⁵ The HEP FORTRAN 77 User's Guide is useful for more specific information on the HEP software.¹⁶

Section III - The Parallel Algorithms

The first step of the research is the application of the parallel processing attributes of the HEP, as described above, to an explicit, one-dimensional Lagrangian hydrocode. An examination of the theory shows that a zone-by-zone parallelization is straightforward. A block-by-block parallelization may be more efficient than the zone-by-zone when the number of processes is significantly less than the number of zones. By a "block" we mean a set of

contiguous zones. The block (or granularity) size is found by dividing the number of zones by the number of processes.

A hydrodynamic simulation is based on the conservation laws of volume, mass, momentum, and energy. In the Lagrangian frame, the corresponding differential equations are differenced on a mesh in which the grid points remain fixed in the material. Let X be the Lagrangian spatial coordinate (the one that identifies the initial location of the mass point) and let μ be the mass coordinate with units of mass per area. The two are related by

$$d\mu = \rho^0 dX \quad (1)$$

where ρ is the mass density as a function of X and the zero superscript indicates ρ is evaluated at time $t = 0$. Let V be the specific volume such that

$$V = \frac{1}{\rho}. \quad (2)$$

This choice of coordinates guarantees conservation of mass.

Assuming rectangular symmetry (slab geometry), the remaining conservation laws in differential form are

- Conservation of volume:

$$\frac{\partial V}{\partial t} = \frac{\partial u}{\partial \mu} \quad (3)$$

- Conservation of momentum:

$$\frac{\partial u}{\partial t} = - \frac{\partial}{\partial \mu} (p + q) \quad (4)$$

- Conservation of energy:

$$\frac{\partial E}{\partial t} = - \frac{\partial}{\partial \mu} [u (p + q)] \quad (5)$$

where u is the velocity (or specific momentum) of a fixed point in the mass, p is the pressure, q is the material viscosity, and E is the total specific energy. In vector notation this system can be written as

$$\frac{\partial U}{\partial t} = \frac{\partial F(U)}{\partial \mu} \quad (6)$$

where

$$U = (V, u, E)^T \quad (7)$$

and

$$F(U) = (u, -(p + q), -u(p + q))^T. \quad (8)$$

As usual, the superscript T denotes the transpose of the array.

If ϵ is the specific internal energy, then

$$E = \epsilon + \frac{1}{2} u^2 . \quad (9)$$

Taking the partial derivative of equation (9) with respect to time and substituting equations (3), (4), and (5) into the result yields an internal energy equation

$$\frac{\partial \epsilon}{\partial t} = - (p + q) \frac{\partial V}{\partial t} \quad (10)$$

in place of equation (5). This system of equations (3), (4), and (5) or (10) is incomplete without an equation for the pressure. The system is closed with a thermomechanical equation of state (EOS) relating p to V and ϵ .

Of the two most common finite differencing schemes, we have chosen the one attributable to von Neumann-Richtmyer¹⁷ over that of Lax-Wendroff. It appears to have good accuracy and less tendency to oscillate in response to strong shocks and rarefactions.¹⁸ For the discretization of the time and mass per area independent variables, we adopt the usual convention by which a superscript denotes time dependence and a subscript indicates the location in the mass variable. With this notation, the half integers denote time and mass centering in the mesh. Thus, the von Neumann-Richtmyer discretization scheme for the conservation laws becomes¹⁹

• Conservation of volume:

$$\frac{v_{j+1/2}^{n+1} - v_{j+1/2}^n}{t^{n+1} - t^n} = \frac{u_{j+1}^{n+1/2} - u_j^{n+1/2}}{u_{j+1} - u_j} \quad (11)$$

• Conservation of momentum:

$$\frac{u_j^{n+1/2} - u_j^{n-1/2}}{t^{n+1/2} - t^{n-1/2}} = - \frac{(p_{j+1/2}^n + q_{j+1/2}^{n-1/2}) - (p_{j-1/2}^n + q_{j-1/2}^{n-1/2})}{u_{j+1/2} - u_{j-1/2}} \quad (12)$$

• Energy equation:

$$\epsilon_{j+1/2}^{n+1} - \epsilon_{j+1/2}^n = - \left[\frac{1}{2} (p_{j+1/2}^{n+1} + p_{j+1/2}^n) + q_{j+1/2}^{n-1/2} \right] (v_{j+1/2}^{n+1} - v_{j+1/2}^n) \quad (13)$$

The EOS expresses $p_{j+1/2}^n$ in terms of $v_{j+1/2}^n$ and $\epsilon_{j+1/2}^n$.

When real viscous effects are negligible or, at least, insufficient to handle severe gradients in the physical properties of the material, an

artificial viscosity is superimposed to average the abrupt variations over several adjacent zones. A well known implementation of artificial viscosity is given by a formula due to von Neumann and Richtmyer as modified by Rosenbluth¹⁷ and Landshoff.²⁰

The superscripts in the difference equations (11), (12), and (13) indicate the order in which they are solved. First, the velocity u is updated from time $t^{n-1/2}$ to time $t^{n+1/2}$ using equation (12) where, in practice, it has not been found necessary to extrapolate the viscosity q to time t^n . Then, in a leapfrog fashion, equation (11) is employed to vault the specific volume V over the velocity from time t^n to time t^{n+1} . Finally, all that remains is the energy equation in which compression and viscous work contribute to the heating of the material.

If the equation of state has a tractable analytic expression for the pressure, it can be substituted into the energy equation (13). Then, upon rearranging terms, the internal energy ϵ is advanced from time t^n to t^{n+1} . One such form is the ideal gas law

$$p = (\gamma - 1) \epsilon \rho \quad (14)$$

where γ is the ratio of specific heats c_V to c_T . A slight generalization of this is the Mie-Grüneisen law

$$p = f(\rho) + \tau \epsilon \rho \quad (15)$$

where $\tau > 0$ is the Grüneisen parameter. This form is often used in research involving shock waves in solids. For both equations of state (14) and (15), it is easy to reduce the von Neumann-Richtmyer implicit discretization of the internal energy equation (13) to an explicit expression for $\epsilon_{j+1/2}^{n+1}$. If the EOS is purely mechanical, meaning p depends only on V , then both the energy equation (13) and the EOS are not needed to complete the system and they may be omitted altogether.

One of the advantages of the parallel computer (MIMD architecture) over the vector computer (SIMD architecture) pertains to the parallelization of the material law routine. These calculations do not in general vectorize. We have observed that, in certain cases, a large portion (often over 75%) of the processing time is spent on computationally intensive material laws.

More complicated situations occur when the EOS is not as straightforward as equation (14) or equation (15) and when energy transport mechanisms such as

conduction become important. In such cases, equation (13) with additional terms for energy transport mechanisms must be solved implicitly.²¹ A discussion of parallel algorithms for implicit equations is contained in the following paragraphs. More general energy equations are beyond the scope of this article. An examination of these problems is a natural extension that the authors would like to promote.

The time and space structure of the data as it has just been described leaves it in a form that is immediately amenable for parallelization. The data structure for the conservation of momentum is illustrated in Figure 2 while the conservation of volume is shown in Figure 3. The discrete conservation of momentum equation (12) has the form

$$u_{\text{new}} = u_{\text{old}} - (\sigma_{\text{right}} - \sigma_{\text{left}}) r \quad (16)$$

where $r = \Delta t / \Delta x$ and $\sigma = p + q$. Similarly, the discrete conservation of volume equation (11) has the form

$$V_{\text{new}} = V_{\text{old}} + (u_{\text{right}} - u_{\text{left}}) r \quad (17)$$

Figures 2 and 3 also show the offsets in both time and space (mass) between the momentum and mass equations as specified by the von Neumann-Richtmyer differencing scheme.

To advance the momentum in time, we construct a subroutine to evaluate equation (16). The main program CREATES the optimal number of copies of this routine and uses the fork-join structure of parallel programming to advance the velocities simultaneously. Asynchronous variables passed between the main program and the copies indicate when the main stream can continue. Then, in similar fork-parallel-join fashion, the new velocities are used to update the zone boundary locations from which new volumes are used to calculate the new energies. These kinds of programming tasks are described further in references [3, page 43] and [15, page 90]. Finally, the explicit equation of state evaluations are done with the same type of zone-by-zone or block-by-block parallel programming structure.

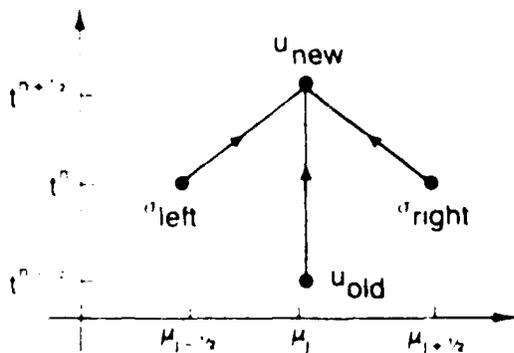


Figure 2. The data structure for the conservation of momentum has an explicit form as stated by equation (16).

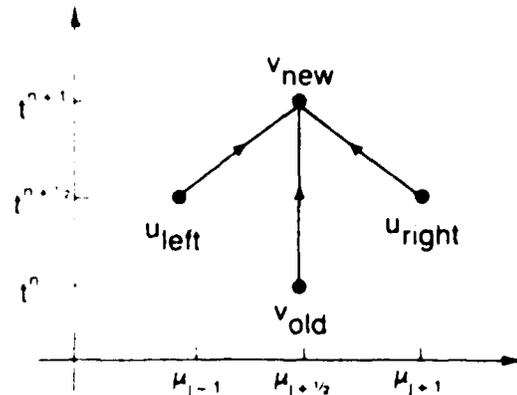


Figure 3. The data structure for the conservation of volume is also explicit as stated by equation (17).

Implicit discretization.

The second stage of the research plan is the parallelization of an implicit scheme. The object of implicit calculations for a hydrocode is the removal of restrictions on the time step. Explicit time steps are usually constrained by the CFL (Courant, Friedrichs, Lewy)^{17,22} condition that is required for stability. Various necessary and sufficient conditions on the CFL number in hydrocodes with classical thermomechanical equations of state such as (14) or (15) are contained in Richtmyer and Morton¹⁷ while some recent results for rate dependent and related computational techniques such as sub-cycling have been developed by Hicks.^{23,24,25} For a general discussion of hydrocode convergence problems, see a couple of the references by Hicks.^{26,27}

The explicit solution advances the components of equation (6) from time $t^{n-1/2}$ to time $t^{n+1/2}$ or from time t^n to time t^{n+1} in terms of quantities evaluated at times t^n or $t^{n+1/2}$, respectively. Such a centered difference scheme yields second order accuracy. However, the time centered quantities have to be evaluated at the forward times $t^{n+1/2}$ or t^{n+1} , respectively, in order for the unconditional stability of an implicit solution to be achieved. A mixture of the two, that is, a weighted average of the explicit and the implicit terms can achieve the stability of the implicit methods while retaining some of the accuracy of the explicit scheme.⁴

The implicit solution of coupled differential equations typically leads to a tridiagonal linear system for which a parallel algorithm is not immediately obvious. It is possible to solve the equations in parallel via an a priori, symbolic inversion of the system.⁴ This involves the evaluation of several recursive sequences. The method divides a recursive sequence into

parallel processes by the concurrent evaluation of even and odd terms (or, more generally, by the concurrent evaluation of the n sequences of terms whose indices are equivalent modulo n). Work on parallel algorithms for the solution of a tridiagonal system will be discussed in a future report. Alternatively, for hydrocode calculations, the original system of differential equations can be divided into uncoupled tridiagonal systems by inserting boundaries along which the values of the unknowns are determined by a stable explicit scheme. Several explicit steps may be required for each implicit integration step.

For the following discussion of the details, assume that a temporal seam of boundary conditions is to be calculated at the mass mesh point u_j . Let the implicit integration step at time t^n be δt . Suppose that the CFL stability condition for the von Neumann-Richtmyer explicit integration requires three time steps to advance the dependent variables from t^n to $t^n + \Delta t$. Let

$$\delta t = \frac{1}{3} \Delta t \quad (18)$$

and use equation (12) to advance $u_{j+i}^{n-1/2}$ from time $t^{n-1/2}$ to time $t^{n-1/2} + \delta t$ for $i = -1, 0, 1$ as shown in Figure 4. Then, since the pressures are known in the appropriate zones up through time t^n , the velocities can be advanced from time $t^{n-1/2} + \delta t$ to time $t^{n-1/2} + 2 \delta t$. At this point, equations (11) and (13) have to be integrated in a similar manner from time t^n to time $t^n + \delta t$ in the surrounding zones. Finally, the velocity is integrated from time $t^{n-1/2} + 2 \delta t$ to time $t^{n-1/2} + 3 \delta t = t^{n+1/2}$ to achieve $u_j^{n+1/2}$ on the internal boundary seam. Figure 4 illustrates the domain of dependence for $u_j^{n+1/2}$. The velocities, densities, pressures, and energies are evaluated explicitly within this domain.

Section IV - Eulerian Coordinates

Once a Lagrangian hydrocode has been constructed, it can be converted to an Eulerian code by making use of the Harlow-Johnson rezoning method.⁵ Figure 5 illustrates the technique. An Eulerian calculation is achieved in two steps. The first is a Lagrangian calculation and the second is a rezoning of the mesh back to its original location. Since the rezoning leaves the mesh points fixed in space, the calculation is Eulerian. Another way of viewing this rezoning scheme is from the point of view of operator splitting. That is, the discretization of the Eulerian operator $\partial/\partial t + u\partial/\partial x$ is split into the

advance of the Lagrangian part ($\partial/\partial t$) followed by the advance of the convective part ($u\partial/\partial x$).

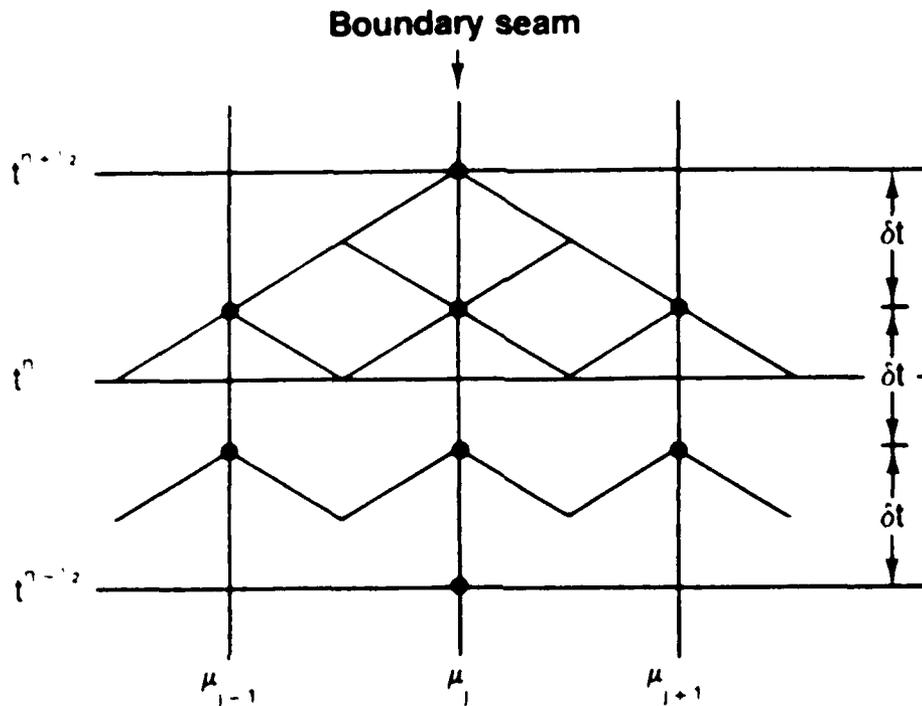


Figure 4. The domain of dependence for the velocities has a half-width of just one zone when only three explicit time steps δt are required to integrate the equations from time $t^{n-1/2}$ to time $t^{n+1/2}$.

In the Lagrangian advance the zone boundaries move from x_j^n to x_j^{n+1} for each $0 < j < J + 1$. The top part of Figure 5 shows the mass moving to the right. Then, from the bottom part of the figure, we see that the rezoning step transfers mass from zone $[x_{j-1}, x_j]$ to zone $[x_j, x_{j+1}]$. Similarly, momentum and energy are transferred from one zone to the next. Careful book-keeping must be maintained to ensure conservation of these quantities.

Implicit here is the assumption that zone boundaries do not travel further than a zone width. That is,

$$x_{j-1}^n < x_j^{n+1} < x_{j+1}^n. \quad (19)$$

If this constraint is violated then the rezoning gets a bit more complicated. The inequality (19) is equivalent to

$$x_{j-1}^n < x_j^n + \Delta t u_j^{n+1/2} < x_{j+1}^n . \quad (20)$$

Thus, if $u_j^{n+1/2} > 0$, for example, then inequality (20) reduces to

$$u_j^{n+1/2} \Delta t < x_{j+1}^n - x_j^n . \quad (21)$$

Constraints of this form are reminiscent of the CFL restriction and are some times referred to as the material-Courant or the mass-flow-Courant restrictions.

This rezoning algorithm automatically conserves volume and mass. Momentum, internal energy, and kinetic energy are each conserved individually according to a step function or a linear interpolation for the accumulated mass per area. The derivation of such models can be found in Hicks and McGrath.²⁸ These lead to recursive linear equations for the dependent variables for which parallel algorithms are under development. For the present, the rezoning calculations are performed within the block implicit framework of the previous section.

Section VI - Timing Results on the HEP

Computer codes have been written for each of the four hydrodynamic simulations: explicit and implicit, Lagrangian and Eulerian finite differences. To demonstrate the improvements in computational efficiency, the results are discussed in this section for the explicit Lagrangian hydrocode with a linear pressure-volume material law. The simple test problem for which the exact solution is known involves shock and rarefaction waves. The initial conditions are an ideal pressure and density shock in the interior of a motionless slab. The boundaries are held fixed and the shock moves forward with a rarefaction proceeding in the opposite direction.²⁹

For the explicit, Lagrangian equations (12) and (13), self-scheduling processes were written and implemented as follows

```
Fork, compute viscosities, join;
Fork, compute velocities, join;
Fork, compute volumes, join; and
Fork, compute pressures, join.
```

For the zone-by-zone algorithm, the self-scheduling within each fork and join means that the processes advance a single zone and then ask for another zone until none remain. For the block-by-block algorithm, between each fork and join, a block of N_B contiguous zones is handed over to each of the N_P parallel processes. The algorithm is pre-scheduled by setting $N_P N_B$ equal to the total number of zones.

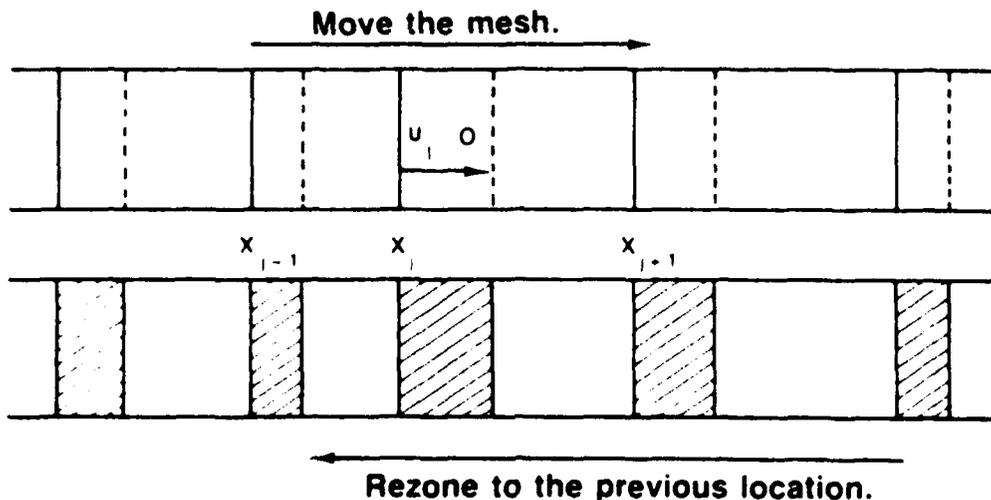


Figure 5. Harlow's method rezones the mesh back to its previous location after each Lagrangian time step.

The zone-by-zone algorithm was run on 21 zones for 4000 integration steps. Figure 6 shows the results over the range of one (a serial computer) to 21 processes. The results for the block-by-block algorithm are shown in Figure 7 for the same variation in the number of processes. It was demonstrated on 4200 zones for 1000 integration steps.

The data shown on Figure 6 can be interpreted according to the following model suggested by discussions in articles by Buzbee,¹ Larson,³⁰ and Flatt.³¹ Consider a computer program that consumes a total processing time of

$$T_t = T_s + T_p \quad (23)$$

where T_p is the total processing time spent on calculations which can be divided into parallel processes. T_s is the processing time required for the calculations that are performed serially when the program is executed in

either serial or parallel mode. Then, up until the point at which the hardware is saturated, the processing time for n parallel processes is approximated by

$$T(n) = T_s + \frac{1}{n} T_p + K T_0(n) \quad (24)$$

where $T_0(n)$ is the CREATE overhead for n parallel processes and K is the number of times that forking and joining occur in the execution sequence. Thus, the last term accounts for the parallel processing overhead, that is, the computer time lost to the creation of the parallel processes and to the communication between them.

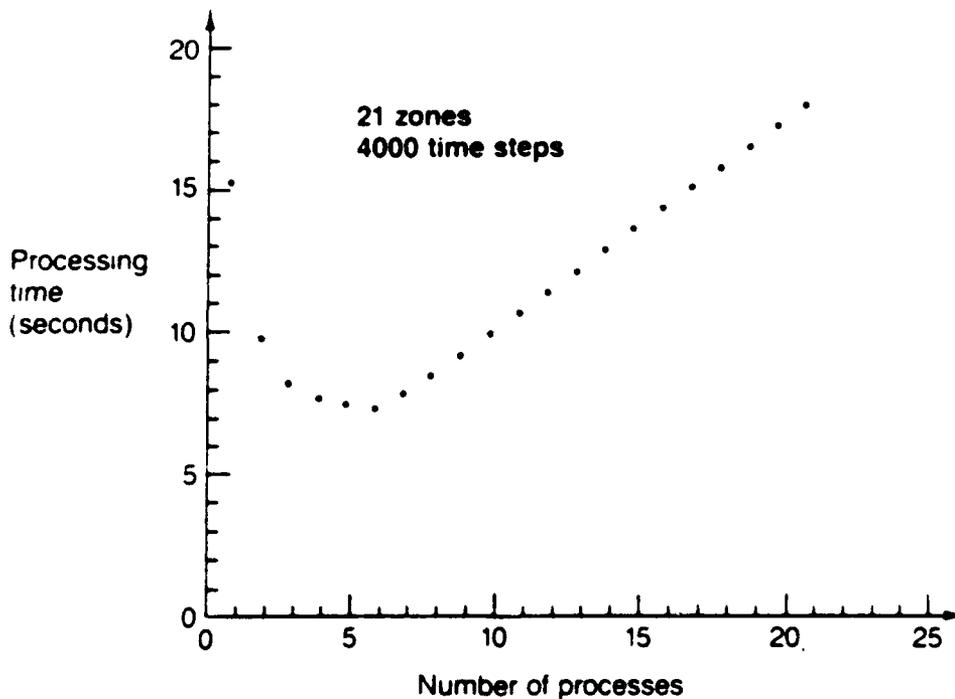


Figure 6. Parallel processing time achieves a minimum and then increases linearly with the number of processes for the zone-by-zone algorithm.

A plot of $T(n)$ versus n yields a curve qualitatively similar to that displayed in Figure 6. In this case, the hardware is saturated at about six processes at which point the term $1/n T_p$ in equation (24) is replaced by a constant. Thus, the effect of the term $K T_0(n)$ is a linear increase in processing time. The zone-by-zone calculations were done with too fine a granularity. That is, the computational chunks were too small. We can increase

their size by going to blocks of zones instead of advancing a single zone at a time. This, of course, leads to the block-by-block parallel structure.

The results of the block-by-block calculations as shown in Figure 7 indicate that the speed-up factor peaks at a value just under nine somewhere in the range of 10-14 processes. The speed-up factor approaches this value because there are essentially nine segments in the calculations: eight in the pipeline plus the store operation. This peaking phenomena in the range of 10-14 processes has been observed by several other investigators.^{2,3,32}

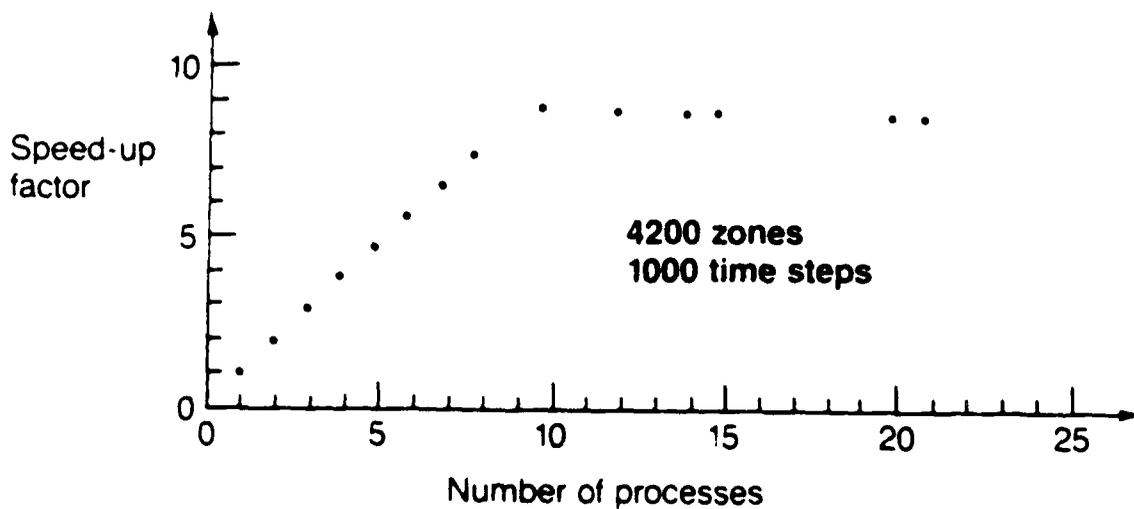


Figure 7. The block-by-block algorithm achieves a speed-up factor of nearly nine at 10 processes before starting to tail off.

Our objective is the maximization of the speed-up factor³¹

$$S(n) = \frac{T_t}{T(n)} = \frac{n}{1 + \frac{(n-1)T_s}{T_t} + \frac{n K T_0(n)}{T_t}} \quad (25)$$

where

$$T_t = T(1) \quad (26)$$

from equation (24). Thus, for $S(n)$ and the efficiency³¹

$$E(n) = \frac{1}{n} S(n) \quad (27)$$

to be high, the denominator of equation (25) has to be small. This means that, for a fixed number of processes n , a quantity limited by the hardware, we need small values of T_s , K , and $T_0(n)$.

The parallel overhead $T_0(n)$ is a machine-dependent quantity. It appears to be proportional to n as shown above. While this is true of other machines,³¹ it can also be proportional to $\log n$.³⁰ In any case, the values of T_s and K are subject to the effectiveness of the parallel algorithms. Thus, for a given machine, it is important to reduce T_s and K as much as possible in order to achieve the highest speed-up factor and the greatest efficiency. One final remark concerns the trade-off between the CREATE statement and the asynchronous variables. The value of K can be altered by reducing the number of CREATEs. The greatest effect can be achieved by eliminating the repeated fork-parallel-join approach employed for both the zone-by-zone and the block-by-block algorithms. To accomplish this, the calculations for viscosities, velocities, volumes, and pressures were incorporated into a single subroutine. The program forks into multiple copies of this routine at the beginning of the simulation. Then each process in either zone-by-zone or block-by-block fashion performs the complete set of calculations for that time step. At the end of the time step, synchronization is achieved through an asynchronous variable, but the process is not terminated by returning to the mainstream. Instead, all processes continue with the next time step. They do not join until the integration is complete.

Of course, the synchronization step constitutes a "join" in the processing to be followed by another "fork". Thus, this approach did not provide the expected improvement in the speed-up factor. Evidently, the memory contention for the asynchronous variable outweighs the savings in CREATE overhead. For these reasons, we believe that Figure 7 illustrates close to the optimum speed-up achievable for the explicit, Lagrangian calculations.

REFERENCES

1. B. L. Buzbee, "The Efficiency of Parallel Processing," Los Alamos Science, 9 (1983).
2. F. J. Smith, "Architecture and Applications of the HEP Multiprocessor Computer System," pp. 241-248 of Proceedings of SPIE--The International Society for Optical Engineering, SPIE 298, Real-Time Signal Processing IV (1981).
3. H. F. Jordan, "Parallel Programming on the HEP Multiple Instruction Stream Computer," Denelcor Report, 20 August 1981.
4. D. L. Hicks, "Parallel processing algorithms for hydrocodes on a computer with MIMD architecture (Denelcor's HEP)," Idaho National Engineering Laboratory Report #EGG-SAAM-6452 (Nov. 1983).
5. D. L. Hicks, "Hydrocodes on the HEP," a chapter in a book on the HEP computer to be published by the MIT press, Ed. J. S. Kowalik (1985).
6. F. H. Harlow, "Hydrodynamic Problems Involving Large Fluid Distortions," J. Assoc. Comp. Mach., 4, 137-142 (1957).
7. D. L. Hicks and R. T. Walsh, "Numerical and Computational Analysis of the Partial Differential Equations in Hydrocodes and Wavecodes," SAND-75-0448 (1976).
8. D. L. Hicks, "Well-Posedness of the Two-Phase Flow Problem, Part 1: Theory and Procedures Developed and Applied to Mechanical EOS with Equal Pressures", Sandia National Laboratory, SAND-79-1435 (1979).
9. D. L. Hicks, "Well-Posedness of the Two-Phase Flow Problem, Part 2: Stability Analyses and Microstructural Models", Sandia National Laboratory, SAND-80-1276 (1980).
10. D. L. Hicks, "Hyperbolic Models for Two-Phase or Two-Material Flow", Sandia National Laboratory, SAND-81-0253 (August 1981).
11. D. L. Hicks, "Microstructural Models for Immiscible Mixtures: Mathematical Modeling Methods," 467-472, Mathematical Modeling in Science and Technology, Eds. X. Avala, R. Kalman, A. Liapis and E. Rodin, Pergamon Press (1984).
12. V. H. Ransom and D. L. Hicks, "Hyperbolic Two Pressure Models for Two Phase Flow," J. of Comp. Phys., 53, 124-151 (1984).
13. D. L. Hicks and M. M. Madsen, "Operator Splitting, Method of Characteristics, and Boundary Value Algorithms," Sandia National Laboratory, SAND-76-0436 (1976).
14. E. L. Lusk and R. A. Overbeck, "Use of Monitors in FORTRAN: A Tutorial on the Barrier, Self-Scheduling DO-Loop, and Ask for Monitors," Argonne National Laboratory Report ANL-84-51 (July 1984).
15. D. J. Evans et al., Parallel Processing Systems, Cambridge University Press (1982).
16. HEP FORTRAN 77 User's Guide, Denelcor Publication No. 9000006, 15 February 1982.
17. R. D. Richtmyer and K. W. Morton, Difference Methods for Initial Value Problems, Interscience (1967).
18. D. L. Hicks and R. Pelzl, "Comparison between a von Neumann-Richtmyer Hydrocode and a Lax Wendroff Hydrocode," AFWL-TR-68-112 (1968).
19. J. F. McGrath, M. J. Dunning, and D. L. Hicks, "Differencing the Momentum Equation in a One Dimensional Hydrocode," KMSF-U1394 (1983).
20. R. Landshoff, "A Numerical Method for Treating Fluid Flow in the Presence of Shocks," LASL Report LA-1930 (1955).
21. M. J. Dunning and J. F. McGrath, "Energy Equation Solutions," KMSF-U1461 (1984).

22. R. Courant, K. O. Friedrichs and H. Lewy, "Uber die Partiellen Differenzengleichungen der Mathematische Physic," Math. Ann., 100, pp. 32-74 (1928).
23. D. L. Hicks, "Stability Analysis of WONDY (A Hydrocode Based on the Artificial Viscosity Method of von Neumann and Richtmyer) for a Special Case of Maxwell's Material Law," Mathematics of Computation, 32, pp. 1123-1130 (1978).
24. D. L. Hicks, "Hydrocode Subcycling Stability", Math. Comp., 37, pp 69-78, (1981).
25. D. L. Hicks, "Stability Concepts Relevant to Discrete Schemes for the Partial Differential Equations of Initial Value Problems", Idaho National Engineering Laboratory Report, AMO-82-6 (1982).
26. D. L. Hicks, "The Hydrocode Convergence Problem -- Part 1," Comp. Meth. in Appl. Mech and Eng., 13, 79 (1978).
27. D. L. Hicks, "The Hydrocode Convergence Problem -- Part 2," Comp. Meth. in Appl. Mech. and Eng., 20, 303 (1979).
28. D. L. Hicks and J. F. McGrath, "Adaptive Grid Techniques for a One-Dimensional Hydrocode," KMSF Report to be published.
29. D. L. Hicks, M. R. Scott and A. H. Treadway, 'Parallel Algorithms for Computational Continuum Dynamics on the HEP, ELXSI, and CRAY-XMP Parallel Processors, Part 1: HEP Results", to appear as a Sandia National Laboratory Report (1985).
30. J. L. Larson, "A Simple Model for Parallel Processing," Computer Magazine, IEEE, pp. 62-69 (July, 1984).
31. H. P. Flatt, "Comments on 'Multitasking on the Cray XMP-2 Multiprocessor,'" Computer Magazine, IEEE, p. 95 (November, 1984).
32. O. M. Lubeck, "Experiences with the Denelcor HEP," presented at the SIAM Conference on Parallel Processing, Norfolk, Virginia (1983).

ATE
LMED
— 8

