Comparisons of Meshless Particle Algorithms and Improvements in Parallelization for the EPIC Code

Prepared by

Gordon R. Johnson
Charles A. Gerlach
Stephen R. Beissel

Southwest Research Institute®
5353 Wayzata Boulevard, Suite 607
Minneapolis, MN 55416

SwRI® Report: 18.17637/09

Subagreement P.O. #2001533539-Mod 17-CLIN 14 (Johns Hopkins University)
Cooperative Agreement No. W911NF-12-2-0022 (Army Research Office)

Technical oversight by

U.S. Army Research Laboratory
Aberdeen Proving Grounds
Aberdeen, MD 21005

Prepared for

Johns Hopkins University
3400 North Charles Street
Baltimore, MD 21218

January, 2019

REPORT DOCUMENTATION PAGE	***Form Approved*** OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 21-01-2019	**2. REPORT TYPE** Final Report	**3. DATES COVERED** *(From - To)* July 2018 – December 2019

4. TITLE AND SUBTITLE Comparisons of Meshless Particle Algorithms and Improvements in Parallelization for the EPIC Code	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER W911NF-12-2-0022
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Gordon R. Johnson, Charles A. Gerlach, and Stephen R. Beissel	**5d. PROJECT NUMBER** 18.17637
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Southwest Research Institute, 5353 Wayzata Boulevard, Suite 607, Minneapolis, MN 55416	**8. PERFORMING ORGANIZATION REPORT NUMBER** 18.17637/09
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Laboratory, Aberdeen Proving Grounds, Aberdeen, MD 21005	**10. SPONSOR/MONITOR'S ACRONYM(S)**
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT

DISTRIBUTION A. Approved for public release: distribution unlimited

13. SUPPLEMENTARY NOTES

.

14. ABSTRACT

This report describes work performed to evaluate and improve the 2018 version of EPIC. The first task includes a thorough evaluation of the three meshless-particle algorithms in EPIC (GPA, Nodal EFG, CPEM). A range of problems is evaluated to consider the effects of the three algorithms with and without conversion of elements into particles, conversion from hexahedral and tetrahedral elements, and a variety of sliding contact interfaces. The second task includes expansion of the initial OpenMP parallelization effort into new algorithms, and examination of potential barriers to scaling problems containing hundreds of millions of elements onto thousands of processor cores.

15. SUBJECT TERMS

EPIC code, meshless particles, impact computations, parallelization, OpenMP, scalability

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Andrew Tonge
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	None	44	**19b. TELEPHONE NUMBER (*Include area code)*** 410-278-1069

Table of Contents

List of Figures

List of Tables

Page

PREFACE

This work was performed at SwRI (Southwest Research Institute) in Minneapolis Minnesota. The funding was provided by ARO (Army Research Office) under the MEDE (Materials under Extreme Dymanic Environments) program. The authors would like to thank Dr. Andrew Tonge from ARL (Army Research Laboratory) and Timothy Holmquist (SwRI) for their contributions to this work.

1.0 SUMMARY

This report describes work performed to evaluate and improve the 2018 version of the EPIC code. The first task includes a thorough evaluation of the three meshless-particle algorithms in EPIC: the Generalized Particle Algorithm (GPA), the Nodal Element-Free Galerkin algorithm (Nodal EFG), and the Combined Particle-Element Method (CPEM). A range of problems is evaluated to consider the effects of the three algorithms with and without conversion of elements into particles, conversion from hexahedral and tetrahedral elements, and a variety of sliding/contact interfaces. The problems include cylinder impact onto a rigid surface, wave propagation through water, expanding spherical shell, large shearing deformations, impact and perforation of a steel projectile onto an aluminum target where there is minimal plastic deformation to the projectile, and impact and perforation of a tungsten projectile onto a steel target where there is significant deformation and fragmentation of both the projectile and the target. A generalized summary is that the GPA is the fastest running, the most robust and the least accurate of the three approaches. The CPEM is the slowest running, less robust than GPA and the most accurate of the three approaches. The Nodal EFG falls in between.

The second task involves the expansion of the initial OpenMP parallelization effort by Tonge, and examination of possible barriers that may stop EPIC from efficiently scaling large problems onto several thousand cores, where the large problems being considered contain hundreds of millions of elements. The 2018 version of EPIC was distributed with most of the initial OpenMP effort. It included the OpenMP directives for the primary element loop, and the loop for GPA particles. The directives for the nodal loop and triangular surface sliding were omitted from the 2018 distribution by accident. This effort has corrected some small problems in the initial effort, and expanded the OpenMP directives to include the primary EFG loop, the primary CPEM loop, and sliding with quadrilateral surfaces. Scaling was tested on a 153 million element problem, and the computations (excluding start-up costs) scale up to at least 2,560 cores.

2.0 INTRODUCTION

There is a continuing need to develop computational capabilities to simulate the responses of weapons and targets to high-velocity impact and explosive detonation. With these capabilities, it is possible to evaluate existing and future designs of weapons and targets, and to aid in the design of improved systems. The capability to perform accurate and efficient computations of these events can result in significant savings in both cost and schedule. This work includes an evaluation of some current meshless-particle capabilities in the 2018 version of EPIC (Elastic-Plastic Impact Computations) [1], and some improvements in parallelization using OpenMP.

The EPIC code is a Lagrangian explicit dynamics code for wave propagation, elastic-plastic flow, material failure, fragmentation, and complex interfaces. It is both a research code and a production code, and it is especially well suited for high-distortion problems caused by intense impulsive loading due to impact and explosive detonation. It contains finite elements, meshless particles, robust contact interfaces and a wide range of material models. It is a FORTRAN code that can be run on a laptop or on large parallel computing systems with thousands of processors. A historical overview of the code and a description of its capabilities are provided by Johnson [1].

The following two chapters describe the evaluation of the meshless-particle capabilities in EPIC, for a range of applications and algorithms; and the expansion of OpenMP parallel directives into more EPIC algorithms, as well as testing the scaling capabilities for problems containing hundreds of millions of elements.

3.0 COMPARISONS OF MESHLESS PARTICLE ALGORITHMS

The EPIC code has a long history of containing meshless particle algorithms. The first particle algorithm was the NABOR algorithm [2] that attempted to allow for variable nodal connectivity by allowing the strains and strain rates to be determined by the neighboring nodes, rather than being attached to elements. This initial approach had many shortcomings, and after the Smoothed Particle Hydrodynamics (SPH) method was expanded for solids by Libersky and Petschek [3], it (SPH) was incorporated into the EPIC code, which also included a first attempt to convert distorted elements into particles [4]. A summary of the SPH work in EPIC is presented in Reference [5].

The SPH options in EPIC were replaced by the Generalized Particle Algorithm (GPA) as it was more accurate than the current SPH algorithms, and was based on a more straightforward formulation. The first GPA publication was in 2000 [6], and the publication containing the current 3D algorithm, with contact interfaces and conversion capabilities, was presented in 2003 [7]. It is a co-location method with all variables (mass, position, velocity, stress, strain etc.) carried at the nodes. It is a strong formulation where the strain rates are determined from a weighted summation of the velocity gradients (between a center node and neighbor nodes within a specified zone of influence), and the forces are determined from a weighted summation of the stress gradients. For a given distribution of neighbor nodes, the three normal strain rates are linearly consistent, but the shear strain rates are not. Like other co-location approaches, tensile instabilities can occur. For conversion, a distorted element is removed and replaced by a particle node, and the particle nodes can contact and slide along surfaces of unconverted elements. The GPA initial conditions can include particles only, particles and elements (where the elements can convert into particles), and elements only (where the elements can convert and slide along the intact element surfaces). Particles of different materials interact through a spring (automatically determined from the material properties) and dashpot between the two particles. This is the fastest running and most robust of the three approaches considered here.

Figure 1 provides a summary of how the conversion process works when elements are converted into particles that use the co-location approach. The left side shows a finite-element mesh before any elements are converted. Note that the red dots represent the centers of the elements where the element variables (stress, strain, etc.) are carried and the gray nodes represent the concentrated (point) masses. The nodes do not have finite diameters, they accept the forces from the element stresses, and carry the positions and velocities. The right side illustrates a conversion of eight elements into GPA particles (represented by gray circles with black dots in the centers). Generally, the conversion occurs after distortion of the element, but the distortions are not shown here for simplicity. During the conversion process, the mass of the converted element is subtracted from the associated nodes and transferred to the particle, the finite element nodes that no longer are attached to any intact elements are discarded, and the converted elements are discarded and their variables are transferred to the associated particle. The particle diameters are determined from the cube root of the volume of the converted element. In addition, the generated particle nodes are attached to one of the adjacent element sides. For the left side of Figure 1, (before conversion) a standard finite-element contact/sliding algorithm [8] can be used as the nodes do not have finite diameters and all of the interactions can be defined as nodes on element sides. Also, on the right side of Figure 1, when the elements are converted to particles the particles do not protrude significantly beyond the initial outline of the finite-element region. After conversion, the particle nodes contact/slide along the sides of the elements, and they also contact one another when overlap occurs between particles of different materials [7].

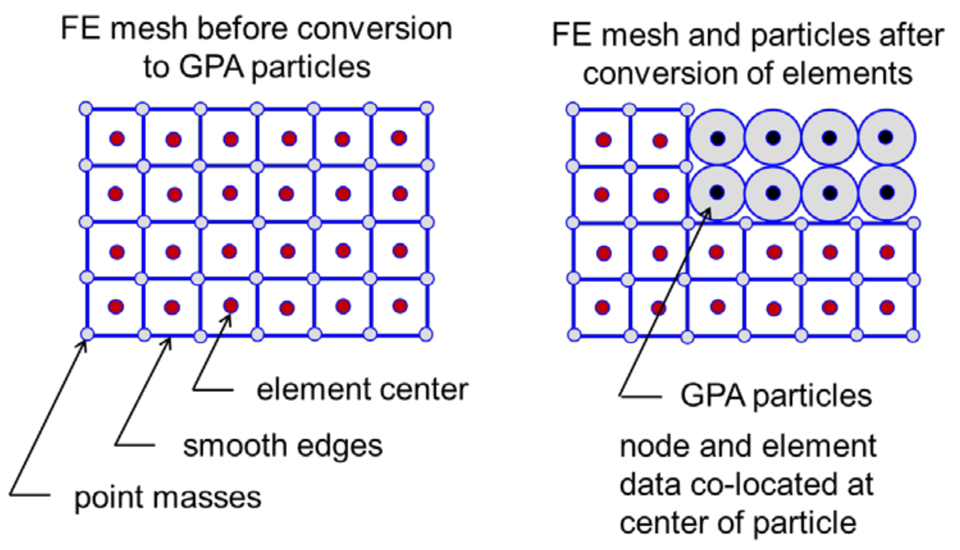
FE mesh before conversion to GPA particles
FE mesh and particles after conversion of elements
element center
smooth edges
point masses
GPA particles
node and element data co-located at center of particle

Figure 1. Description of the approach for conversion of elements into particles for the co-location algorithm used by the GPA. The neighbor nodes for the GPA particles are limited to the other GPA particles. The neighbor nodes are used to determine strains, strain rates and nodal forces.

A Combined Particle-Element Method (CPEM) was presented in 2015 [9], and it carries the mass, positions and velocities at the nodes, but the stresses, strains, and other state variables are carried at massless stress points. Here the strain rates and nodal forces are determined by a weak-form Element-Free Galerkin (EFG) algorithm, which provides increased accuracy, but longer computing times and less robustness. For a given distribution of neighbor nodes, all six strain-rate components are linearly consistent, which meets the patch test. For conversion, there is no replacement of an element by a particle node, but rather the strain rates for the stress point simply change from a finite-element formulation to an EFG particle algorithm. There are more options for sliding and contact because the mass is carried on the nodes, and they have finite diameters. One option is to simply have particle-to-particle contact with no sliding interfaces, and this option can be used with both unconverted stress points (elements) or converted stress points. Another option is for the special case when only one material is converted, and this condition uses the same particle-element sliding/contact algorithm as used for the GPA (and Nodal EFG). The third option is to use a specific algorithm for CPEM when multiple materials are converted [10]. This option is more complex and less robust, and can require increased computing times.

Figure 2 provides a summary of the conversion process when elements are converted into particles using the CPEM. The left side shows a finite-element mesh before any elements are converted. The red dots represent the centers of the elements where the element variables (stress, strain, etc.) are carried, and these are called stress points. The gray nodes accept the forces from the stress points, and they carry the mass, positions and velocities. Unlike the GPA, these mass nodes do have finite diameters that represent the volume required for the mass and density of the material that contributes to that node, and it is this characteristic that causes issues associated with contact/sliding during conversion. Again, the diameters of the mass nodes are equal to the cube root of the volumes, and the volumes are simply the nodal mass divided by the density of the material contributing to the nodal mass. The right side illustrates a conversion of eight elements into the CPEM particle algorithm, but again the distortions are not shown here for simplicity. Unlike the GPA conversion process, no nodes or elements (stress-points) are deleted. Rather the element (stress point) variables are computed (after conversion) from the existing neighbor nodes using an Element-Free Galerkin (EFG) approach [11], which is consistent and does not experience tensile instabilities. The converted stress points are also given a diameter (cube root of the volume) as this is used to determine influence distances to the neighbor mass nodes (in dimensionless diameters). Another illustration is provided in Figure 3, where a cylinder impacts a rigid surface. It simply shows how the black stress points pick up a set of neighbor nodes that are different from the nodes attached to the finite element before conversion.

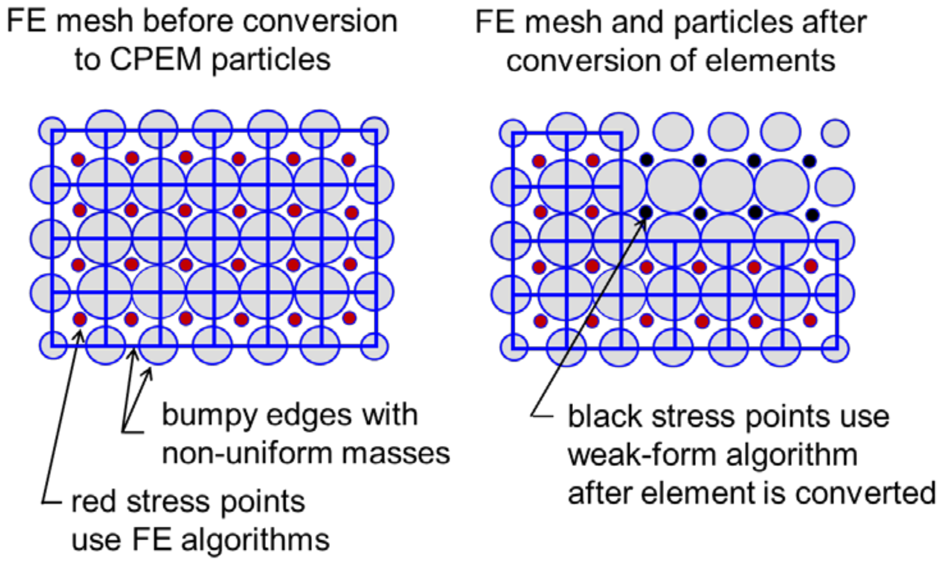
FE mesh before conversion to CPEM particles
FE mesh and particles after conversion of elements
bumpy edges with non-uniform masses
red stress points use FE algorithms
black stress points use weak-form algorithm after element is converted

Figure 2. Description of the approach for conversion of elements into particles for the CPEM, which uses stress points and mass nodes.

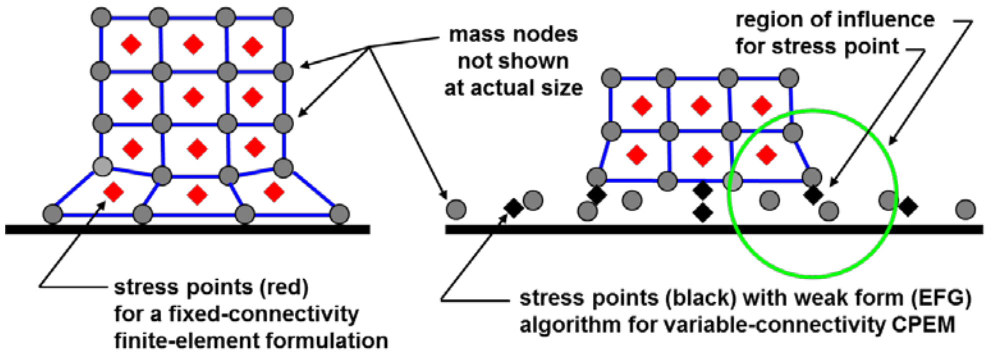
mass nodes not shown at actual size
region of influence for stress point
stress points (red) for a fixed-connectivity finite-element formulation
stress points (black) with weak form (EFG) algorithm for variable-connectivity CPEM

Figure 3. Illustration of the CPEM for a cylinder impacting a rigid surface. Red stress points (with lower strains) used fixed-connectivity finite-element algorithms, and black stress points (with higher strains) use the variable-connectivity EFG particle algorithm.

The Nodal EFG algorithm with conversion is new and has not yet been published in the literature. However, the basic Nodal EFG algorithm was presented in 1996 by Beissel and Belytschko [12]. It identified the presence of zero-energy modes and they were reduced by a perturbation approach. It is a co-location algorithm (similar to GPA), but it is based on a weak (EFG) formulation which is more accurate than the GPA. It has a similar conversion algorithm (replacement of an element by a particle), but the center node includes neighbor nodes from both converted particles and interior nodes in the unconverted element regions, as shown in Figure 4. This allows for smoother transitions from distorted elements to converted particles, as there is generally a good neighborhood of nodes to determine the strain rates and forces. The contact and conversion algorithms are similar to those of the GPA. An SPH bond viscosity is used to reduce the zero-energy modes. Although this approach is new, it appears that it is more accurate than GPA and less accurate than CPEM. It takes more computing time than GPA, but less than CPEM. It is also less robust than the GPA.

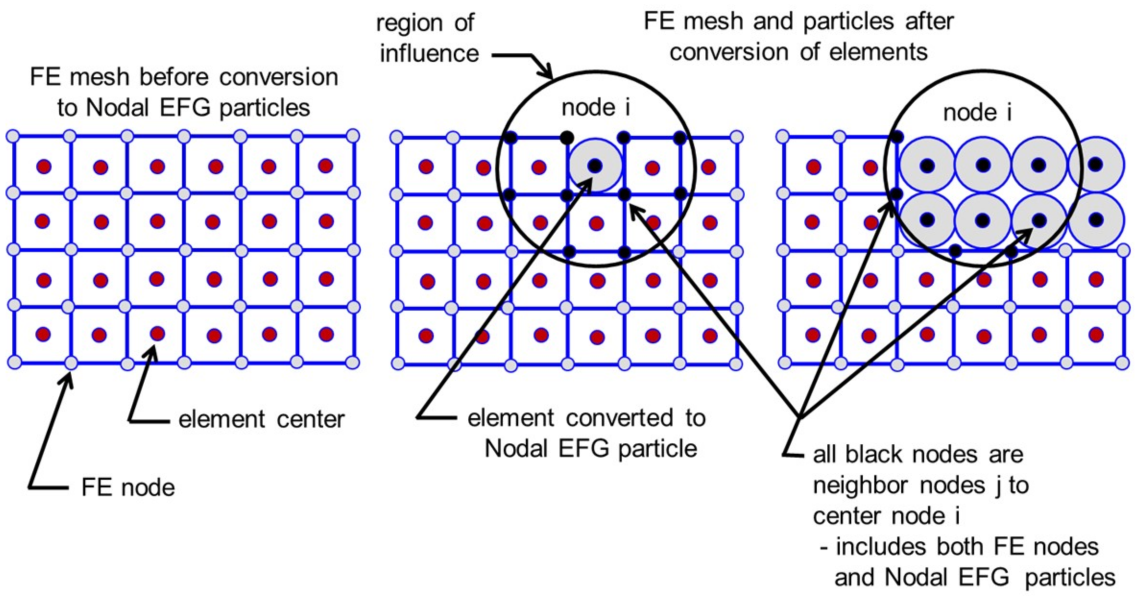
region of influence
FE mesh and particles after conversion of elements
FE mesh before conversion to Nodal EFG particles
node i
node i
element center
FE node
element converted to Nodal EFG particle
all black nodes are neighbor nodes j to center node i
- includes both FE nodes and Nodal EFG particles

Figure 4. Description of the approach for conversion of elements into particles for the co-location algorithm used by the Nodal EFG approach. The neighbor nodes for the Nodal EFG particles are not limited to the other EFG particles, but also include finite-element nodes within the region of influence. The neighbor nodes are used to determine strains, strain rates and nodal forces.

In the subsections that follow, several different problems of interest are considered. Included are computations using the three particle algorithms described previously, conversion with each of these three algorithms, and results using an initial mesh consisting of hexahedral elements and of tetrahedral elements. Some different sliding/contact algorithms are also considered. Additional comments concerning the three algorithms are made in evaluating and comparing the results, but a detailed description of the algorithms must be obtained from the references noted previously.

Before presenting the computed results a few comments should be made regarding hexahedral and tetrahedral elements, as there are some differences between these two elements when converting from elements to particles. The top portion of Figure 5 shows two cubes, one containing 27 hexahedral elements and one containing 24 tetrahedral elements. For all of the problems considered here each group of 24 tetrahedral elements corresponds to 27 hexahedral elements. This does not necessarily mean that comparable accuracy is obtained with these two meshes even though the number of elements is almost equal. A difference is that the hexahedral mesh contains one node for every element (for an infinite media), whereas the tetrahedral mesh, contains approximately one node for every five elements. Therefore, for an equal number of elements the hexahedral mesh has five times more nodes and for an equal number of nodes the tetrahedral mesh has five times more elements. For equivalent accuracies, there is probably an intermediate condition where there are more nodes for the hexahedral mesh and more elements for the tetrahedral mesh. For the problems presented here, the hexahedral meshes are clearly more refined than the tetrahedral meshes (slightly more elements and many more nodes).

There are also some significant differences when the hexahedral and tetrahedral elements are converted to particles, as shown in the bottom portion of Figure 5. The hexahedral mesh generates a uniform distribution of particles with no overlap, whereas the tetrahedral mesh generates a non-uniform distribution of particles with significant gaps and overlapping particles.

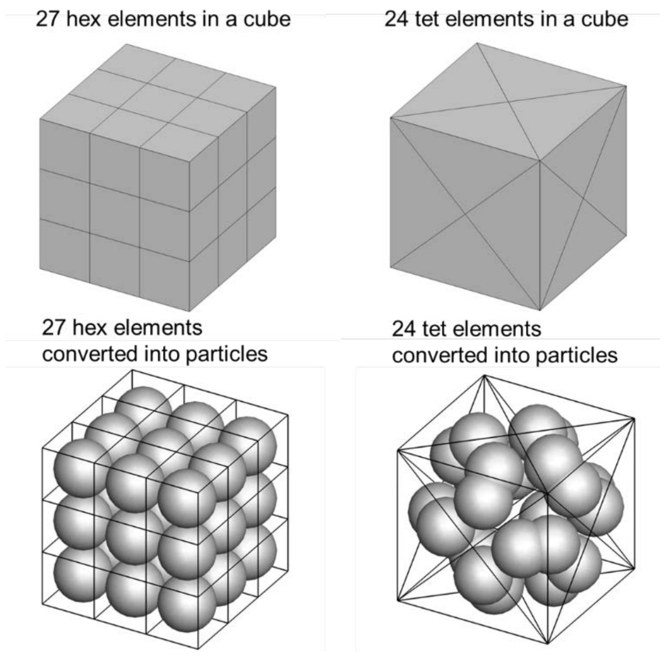
27 hex elements in a cube
24 tet elements in a cube
27 hex elements converted into particles
24 tet elements converted into particles

Figure 5. Element and particle arrangements for hexahedral and tetrahedral elements.

The following computations were performed with the 2018 version of EPIC [1]. All of the computations were run in 3D geometry with a plane of symmetry, and with a sound speed fraction (SSF) of 0.7, a bond viscosity fraction (QFACT) of 0.5, a region of influence of 1.8 particle diameters, and a conversion strain of 0.3 (with the exception of the wave propagation problems). They were all run in a serial mode such that run times could be determined and compared in a straightforward manner. The first four sets of computations are included to evaluate the various approaches by comparing to finite element results, as particle methods are not required for these computations. The last two sets of computations demonstrate the use of particle methods for highly distorted applications that cannot be accurately computed with finite element methods.

3.1 Cylinder Impact Computations

The first set of computations is for an Armco iron (EPIC library material #6) [13] cylinder impacting a rigid surface as shown in Figures 6 and 7. The cylinders have a length of 6.0 cm, a diameter of 4.0 cm, and an impact velocity of 200 m/s. The nodes are attached to the interface.

In Figure 6, it is assumed that the finite-element computation provides the correct result, with the cylinder represented by 34,020 hexahedral elements and 37,345 nodes. The number of particles is equal to the number of elements for the particle-only computations, and $L^* = L_{final}/L_0$ and $D^* = D_{final}/D_0$. The CPU times provide an indication of the run-time efficiencies, but the number of cycles also affect the CPU times. All of the particle runs take significantly more CPU time than the finite element computation. It should be emphasized that these computations are run with the various particle approaches such that they can be compared to an accurate (finite-element) computation. There would be no logical reason to use particle approaches for a cylinder-impact computation where the strains could be accurately handled with finite elements. Another general observation is that all of the conversion computations take significantly less CPU time than the comparable particle-only computations.

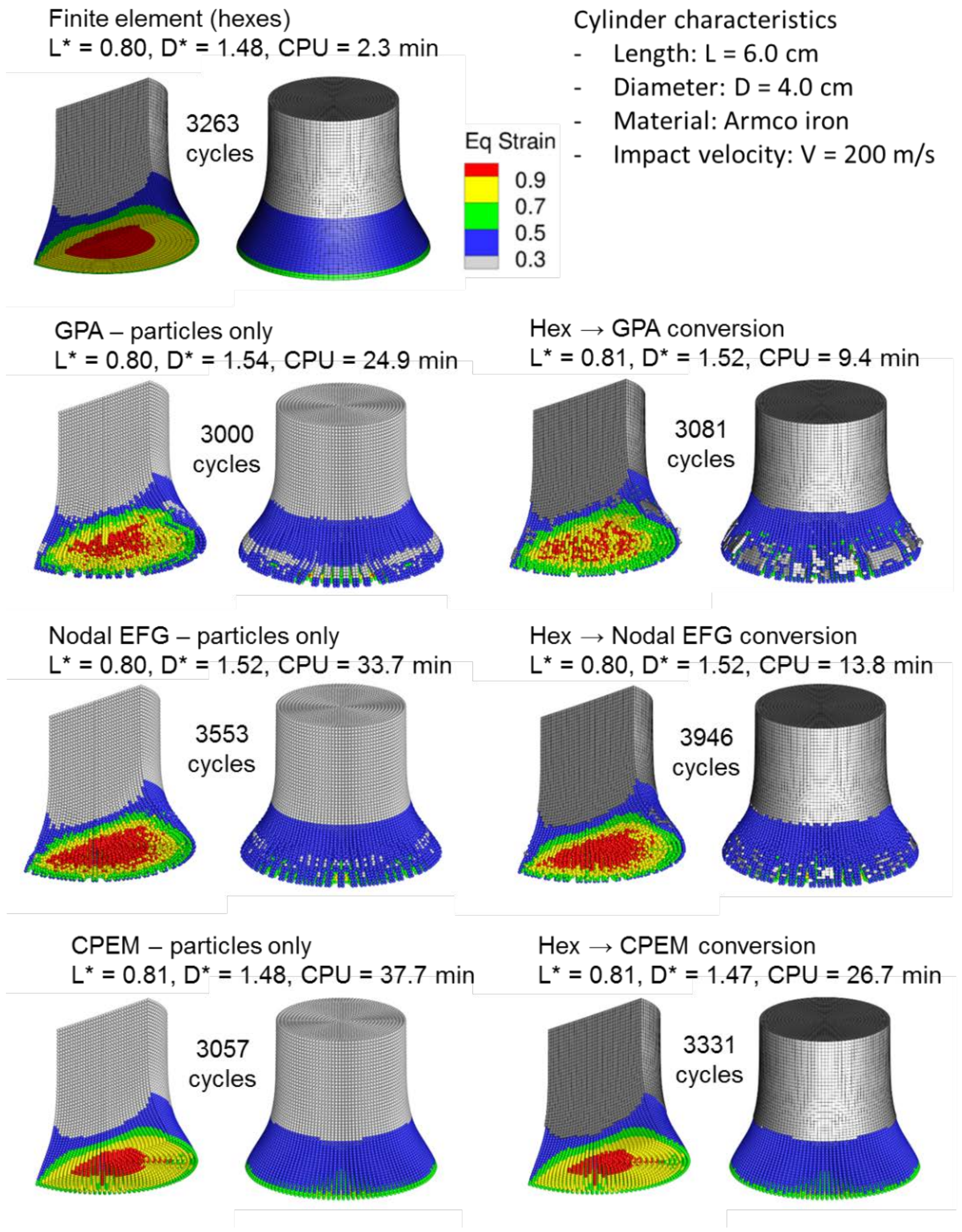
Finite element (hexes)
L* = 0.80, D* = 1.48, CPU = 2.3 min
3263 cycles
Eq Strain
0.9
0.7
0.5
0.3
Cylinder characteristics
- Length: L = 6.0 cm
- Diameter: D = 4.0 cm
- Material: Armco iron
- Impact velocity: V = 200 m/s
GPA – particles only
L* = 0.80, D* = 1.54, CPU = 24.9 min
3000 cycles
Hex → GPA conversion
L* = 0.81, D* = 1.52, CPU = 9.4 min
3081 cycles
Nodal EFG – particles only
L* = 0.80, D* = 1.52, CPU = 33.7 min
3553 cycles
Hex → Nodal EFG conversion
L* = 0.80, D* = 1.52, CPU = 13.8 min
3946 cycles
CPEM – particles only
L* = 0.81, D* = 1.48, CPU = 37.7 min
3057 cycles
Hex → CPEM conversion
L* = 0.81, D* = 1.47, CPU = 26.7 min
3331 cycles

Figure 6. Cylinder impact results for hexahedral elements only, particles only, and conversion of elements to particles. Includes GPA, Nodal EFG and CPEM.

The two GPA computations are the least accurate, but they also require the least amount of CPU time. The particles around the circumference at the interface tend to clump into pairs (under tension), and this is a characteristic of the co-location algorithm. The GPA is also the most robust of the three algorithms, as will be demonstrated in subsequent problems.

The two Nodal EFG computations are more accurate than the GPA computations, but less accurate than the CPEM computations. As this is a co-location algorithm, the particles around the circumference at the interface also tend to clump into pairs. This algorithm is not as robust as the GPA as a matrix inversion is required to determine the strain rates and forces. A tolerance is input for the inversion, but it is not a clearly defined value that works for all problems. A higher tolerance skips over more particles that do not have an adequate neighborhood, than does a lower tolerance. These cylinder-impact problems all have good neighborhoods so this is not an issue. For highly distorted applications, where fragmentation occurs, this is more of an issue.

The two CPEM computations are the most accurate and require the most CPU time. The CPEM is not as robust as the GPA and it also requires a tolerance for a matrix inversion. In addition, it sometimes requires a more complex sliding/contact algorithm than the other two approaches. A desirable feature of the CPEM is that there is no deletion of elements and addition of particles as the solution progresses. Rather, when the conversion strain is reached the strain-rate and force algorithms simply change from an element formulation to an EFG particle formulation. This same feature allows the conversion to be made at larger strains than can be used for the other algorithms. This positive characteristic for CPEM is not included in this study as a maximum conversion stain of 0.3 is used to be compatible with the GPA and Nodal EFG approaches.

In Figure 7, conversion computations are provided for a cylinder initially composed of tetrahedral elements. It is represented by 30,240 tetrahedral elements and only 7,071 nodes. As noted previously (in Figure 5) the number of elements is 24/27 times the number of hexahedral elements, and the number of nodes is much less than the number of nodes for the hexahedral mesh. Although the finite-element results for the hexahedral and tetrahedral meshes are almost identical, the initial tetrahedral mesh is coarser than the hexahedral mesh and generalizations regarding the accuracy of the two elements cannot be made from these computed results. All of these computations run faster than the comparable hexahedral meshes in Figure 6, and this is due to the coarser initial mesh and a tetrahedral-element formulation that is less complex than the hexahedral-element formulation. The trends for the tetrahedral-based results in Figure 7 are similar to the trends for the hexahedral-based results in Figure 6.

It is of interest to look at time-history responses of a point in the cylinder to determine the agreement between the various approaches, and to determine if there are sudden transitions when the element is converted into a particle. Figures 8-10 show pressure, von Mises equivalent stress, and equivalent plastic strain, as a function of time for the three particle-only formulations (Figure 8), conversion from hexahedral elements (Figure 9) and conversion from tetrahedral elements (Figure 10). The location of the monitoring point is taken at half the radius (R = 1.0 cm) and a quarter of the distance from the bottom to the top of the cylinder (Z = 1.5 cm). For all cases, a finite-element solution is provided for comparison purposes, and it is taken to be the correct result.

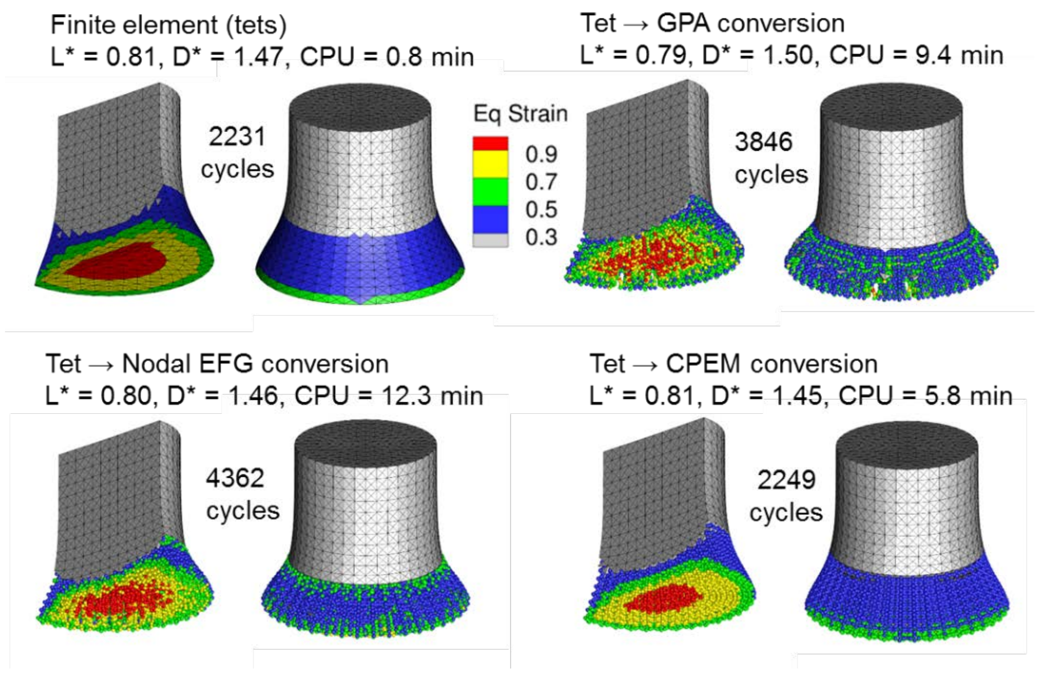
Finite element (tets)
L* = 0.81, D* = 1.47, CPU = 0.8 min
2231 cycles
Eq Strain
0.9
0.7
0.5
0.3
Tet → GPA conversion
L* = 0.79, D* = 1.50, CPU = 9.4 min
3846 cycles
Tet → Nodal EFG conversion
L* = 0.80, D* = 1.46, CPU = 12.3 min
4362 cycles
Tet → CPEM conversion
L* = 0.81, D* = 1.45, CPU = 5.8 min
2249 cycles

Figure 7. Cylinder impact results for tetrahedral elements only, and conversion of elements to particles. Includes GPA, Nodal EFG and CPEM.

For the three particles algorithms in Figure 8, the CPEM clearly provides the most accurate results followed by the Nodal EFG and GPA approaches. Both the GPA and Nodal EFG approaches accurately represent the first compressive and tensile pressure peaks, but they drop off significantly at the later times. The GPA drops off more than the Nodal EFG approach. The decreases in the equivalent stress at about 100 μs occur as the plastic flow is completed and the material becomes elastic.

For the three conversion (from hexahedral elements) results in Figure 9 the same trends are evident, except the results for the GPA and the Nodal EFG approach are more accurate than for the particles-only results in Figure 8. This is because more of the solution is performed in the more accurate finite-element formulation. The conversions take place at an equivalent strain of 0.3 and this occurs at about 40 μs. There do not appear to be any sudden transitions in the three variables at the time of conversion.

The three conversion (from tetrahedral elements) results are shown in Figure 10, and they are compared to a tetrahedral element computation. Although no direct comparison is made between the hexahedral and tetrahedral element results, they are essentially identical except the tetrahedral-element result does not attain as high a pressure peak as the hexahedral-element result. This is most likely due to the coarser mesh used for the tetrahedral-element computation.

It can be seen in Figure 10 that the GPA experiences a significant transition after the conversion. A more detailed examination of many particles indicated that more oscillations occurred with the GPA, some by Nodal EFG, and very few with CPEM. Also, more oscillations occurred when conversion occurred from tetrahedral elements than from hexahedral elements. These oscillations are essentially inaccuracies, rather than instabilities, and the total energy for all of the cylinder-impact problems was accurately conserved. It will be shown in the next (wave propagation) problem that there are issues with the accuracy of the pressures when the particles are generated from a tetrahedral arrangement of elements.

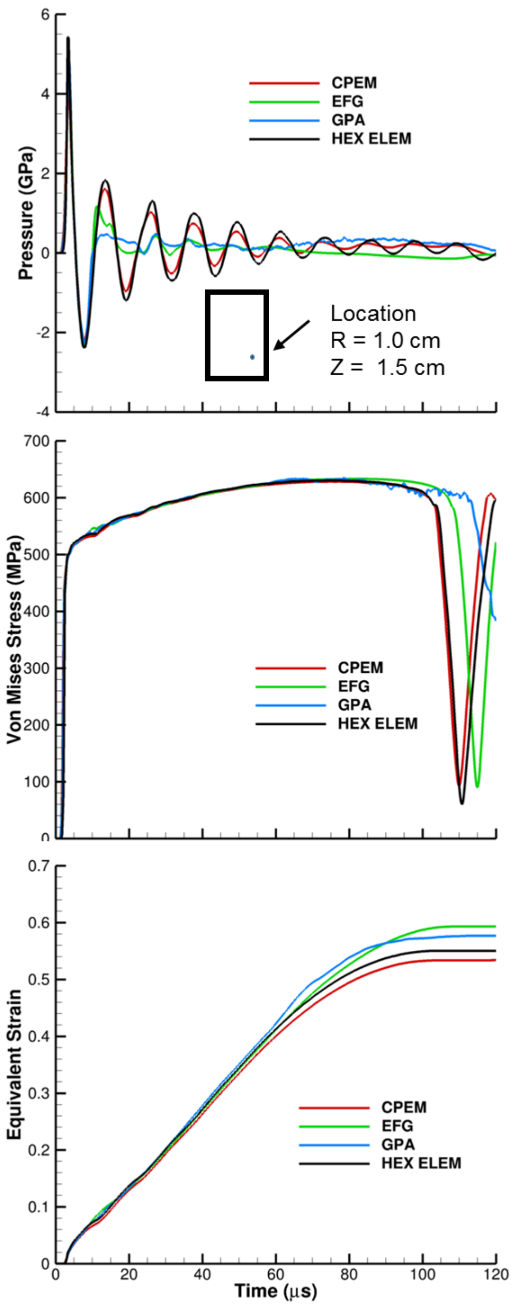

Figure 8. Time-history responses for a point in the cylinder for the three particle algorithms and a hexahedral finite-element formulation.

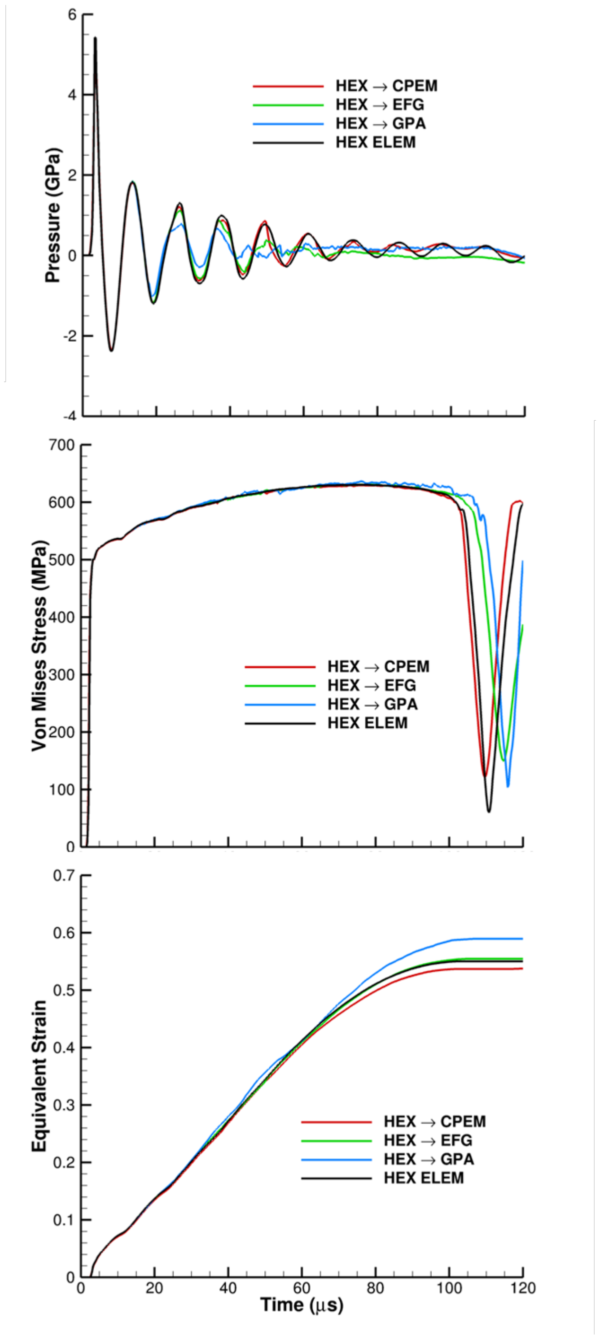

Figure 9. Time-history responses for a point in the cylinder for conversion of hexahedral elements into the three particle algorithms, and for a hexahedral finite-element formulation.

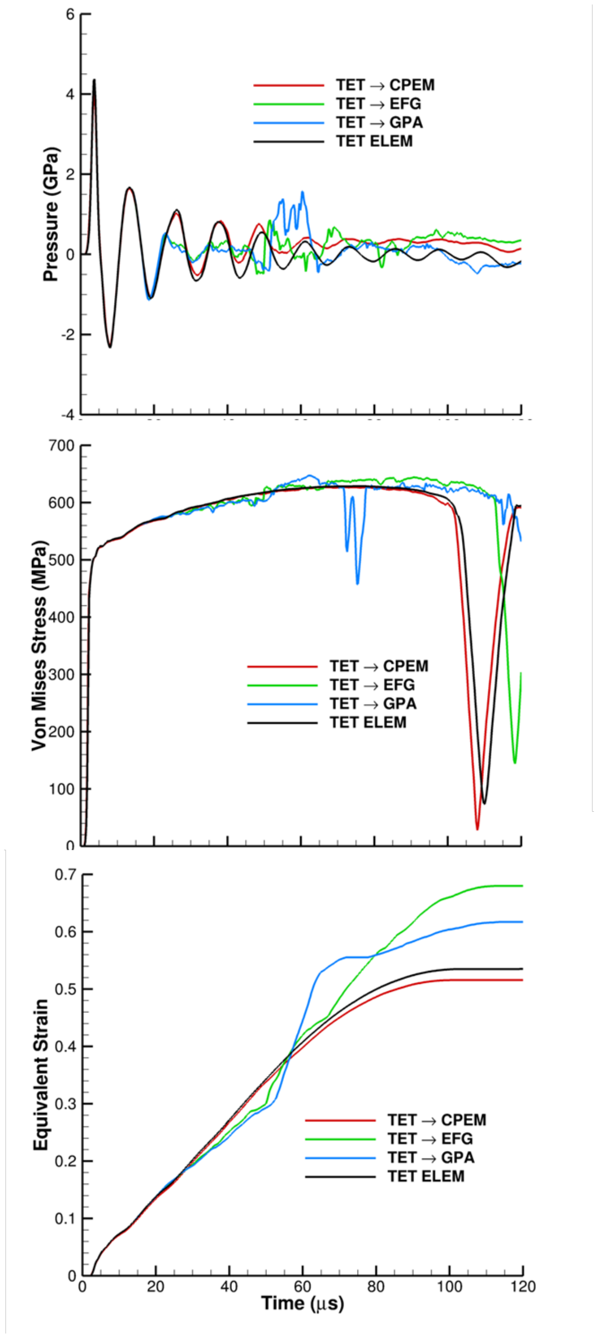

Figure 10. Time-history responses for a point in the cylinder for conversion of tetrahedral elements into the three particle algorithms, and for a tetrahedral finite-element formulation.

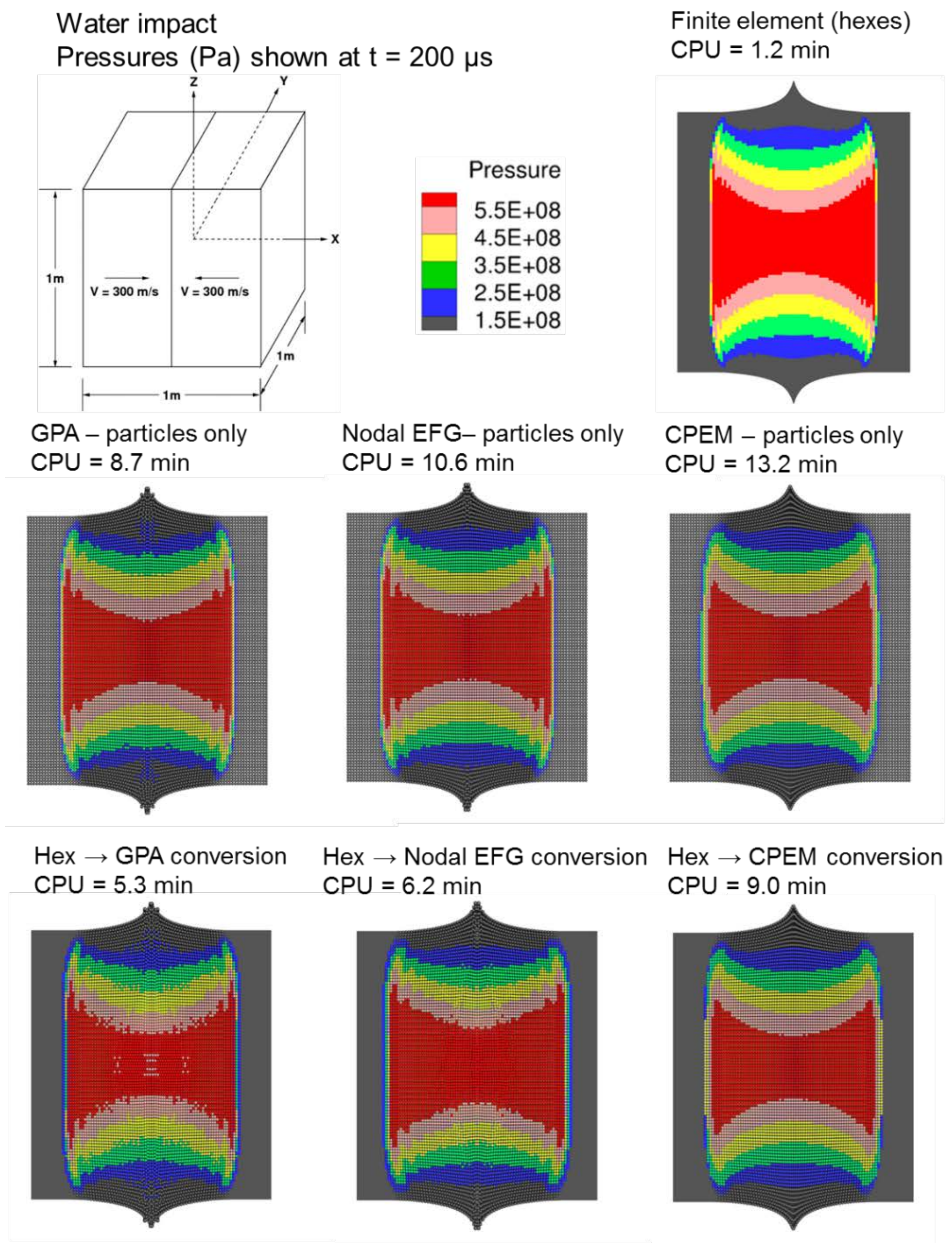

Figure 11. Pressure contours for the water wave-propagation problem with particles only and particles converted from hexahedral elements.

3.2 Wave Propagation Computations

The second set of computations includes wave propagation, as shown in Figure 11. This problem consists of two bodies of water (EPIC library material # 13) impacting at 300 m/s. This impact produces a strong shock with a compressive volumetric strain of 0.169, a maximum pressure of 0.624 GPa, and a wave propagation velocity of 2080 m/s (much higher than the acoustic sound velocity of 1480 m/s) [9]. Again, this problem is considered because the results can be compared to those of a finite-element computation to assess the accuracy, and there would be no logical reason to use particle approaches for this problem. The three particle-only results (center row) provide good agreement with the finite-element result, although the finite-element result (with hexahedral elements) has a slightly sharper wave front because the particle algorithms have a larger region of influence (which tends to spread the wave front).

For the conversion results (bottom row) the conversion strain is reduced to 0.04 (about half that which exists in the fully shocked region), and this causes the conversion to occur on the leading edge of the shock wave. Also, the particles were arranged in a hexahedral arrangement as shown in Figure 5. All three solutions are in good general agreement with the finite-element solution, with the CPEM being the most accurate, followed by the Nodal EFG and the GPA solutions. Again, the GPA requires the least CPU time, followed by the Nodal EFG and the CPEM.

This same problem was run with tetrahedral elements and the results were essentially identical to those obtained with hexahedral elements. However, when the conversion runs were performed, all three particle algorithms failed to produce the correct pressures. Apparently, when the tetrahedral elements were converted, the particle arrangements were similar to those shown in Figure 5, with significant overlaps and open spaces. For the GPA and Nodal EFG computations the particles simply collapsed into the open spaces and the volumetric strains (and associated pressures) were not computed accurately. The CPEM computations provided better results, but the pressures had significantly greater oscillations than the results in Figure 11. Additional computations were performed at different impact velocities, and with different materials, and it appears that the pressures generated from a tetrahedral element arrangement tend to be too low, especially for the GPA and Nodal EFG algorithms. It should be noted, however, that a low conversion strain of 0.04 was used here, and a larger conversion strain of 0.30 might tend to reduce these inaccuracies.

3.3 Expanding Spherical Shell Computations

The third set of computations involves a spherical shell subjected to an outward radial velocity. This problem is included to illustrate the capabilities of the various approaches for tensile pressures and strains. Again, the material is Armco iron and material failure is not allowed. Any failures are due to the numerical algorithm, not the material failure model. The initial geometry and finite element results are shown in Figure 12. The spherical shell is given an initial outward velocity of 250 m/s, and the outer and inner diameters were selected to provide nearly cubical hexahedral elements in the center of the thickness of the shell. The number of elements is slightly higher for the hexahedral mesh, but the number of nodes is significantly higher (as discussed previously). Both of the finite element results provide essentially the same results and the deformed shapes hold their spherical geometry. Results are shown at t = 250 μs, when the plastic deformation is essentially completed.

Figure 13 shows results for computations that use particles only. The results on the left are when the particles are arranged in a hexahedral element formulation, with one particle for each hexahedral element, as shown on the left side of Figure 12; and the results on the right are when the particles are arranged in a tetrahedral arrangement, as shown on the right side of Figure 12. A similar arrangement of results is shown for conversion computations in Figure 14, where the particles are converted from a hexahedral mesh (left) and a tetrahedral mesh (right).

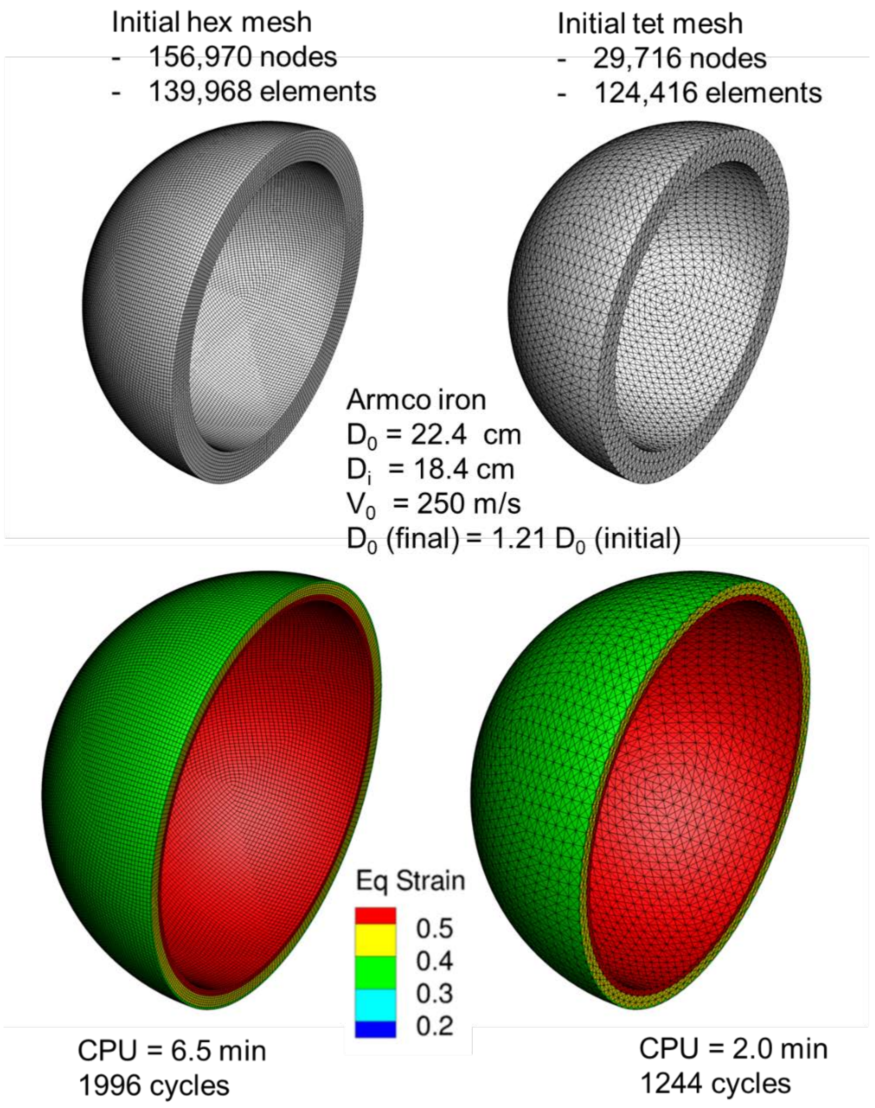

Figure 12. Expanding spherical shell results for hexahedral elements and tetrahedral elements. Final results (equivalent plastic strain) shown at t = 250 μs.

In Figure 13, it can be seen that there are significant numerical fractures for both of the GPA computations (top) and for the Nodal EFG computation in the hex configuration (left center). The Nodal EFG computation in the tet configuration (right center) does not show major numerical fractures, but the plastic strain results are not accurate. The two CPEM results (bottom) are in good agreement with the finite element results. The numerical fractures in the GPA and Nodal EFG computations occur when adjacent particles clump together and lose touch with the other neighbor nodes on the other sides. After the particles lose their neighbors they cannot develop any additional strains, and that is why the strains are much lower than the correct strains, even though the spheres are expanded to a greater radius than the finite element results. A detailed discussion of tensile instabilities for Smoothed Particle Hydrodynamics (SPH), also in a co-location framework, is provided by Swegle et al. [14].

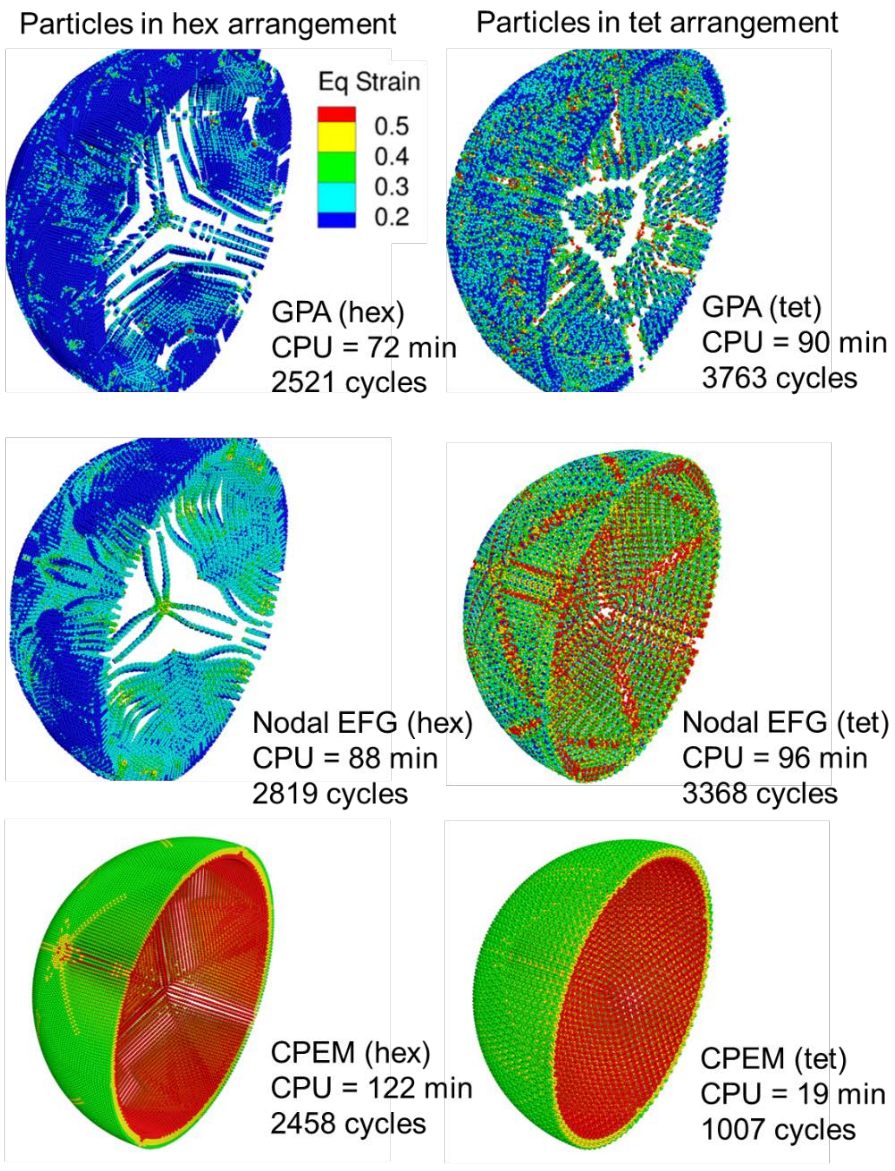

Figure 13. Expanding spherical shell results for GPA, Nodal EFG and CPEM algorithms (particles only). Left side is for particles arranged in hexahedral configuration and right side is for particles arranged in tetrahedral configuration. Final results (equivalent plastic strain) shown at t = 250 μs.

Figure 14 shows results for conversion from hexahedral elements (left) and tetrahedral elements (right). Here the results for GPA and Nodal EFG are better than those shown in Figure 13, and this is because the initial deformations occur in the elements, which are more accurate than the particles. It is interesting that the results appear to be better for the tetrahedral arrangements, both for particles only and for the conversion. This is in contrast to the previous wave propagation computations in Figure 11, and the reason for this difference is not obvious.

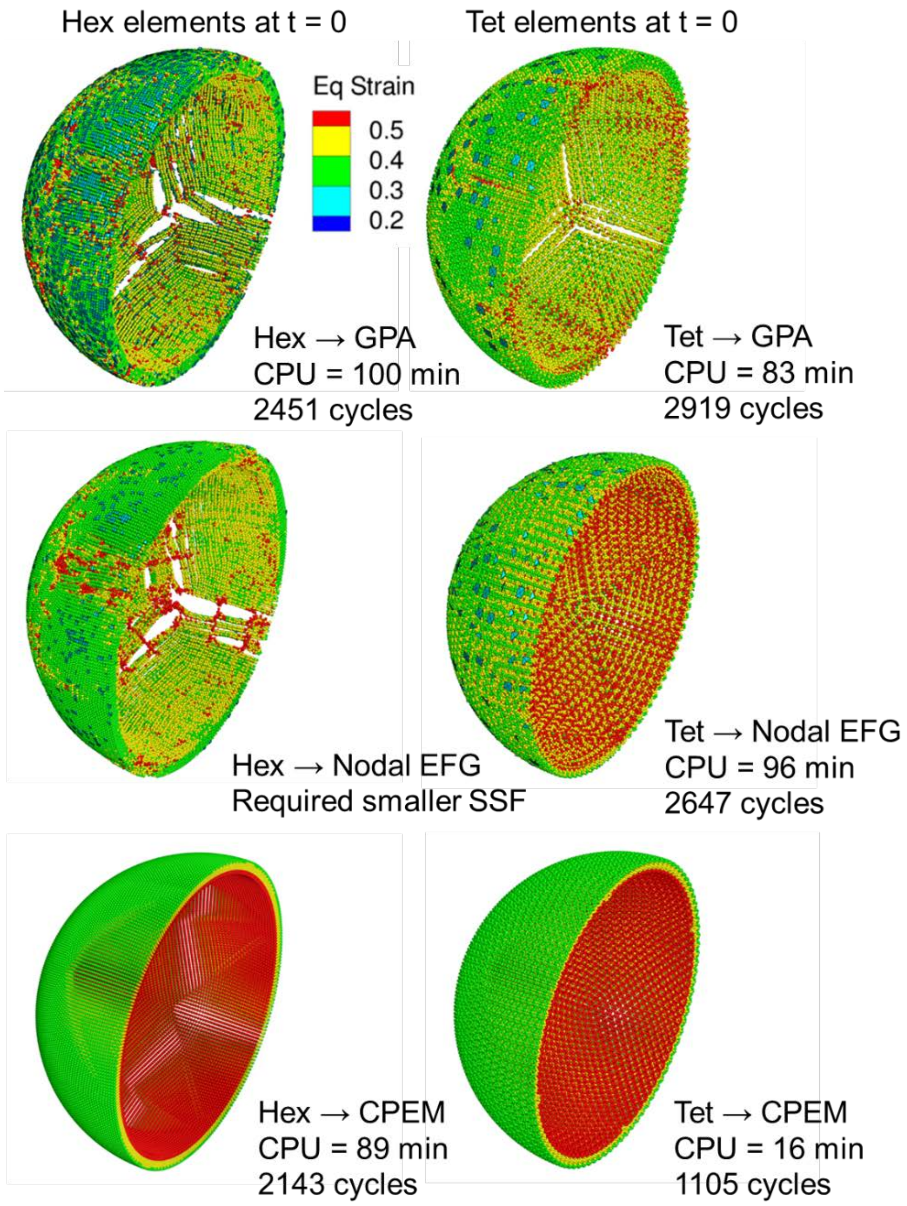

Figure 14. Expanding spherical shell results for GPA, Nodal EFG and CPEM particles converted from elements. Left side is for particles converted from hexahedral elements and right side is for particles converted from tetrahedral elements. Final results (equivalent plastic strain) shown at t = 250 μs.

3.4 Shear Deformation Computations

The fourth set of computations involves shearing deformations imposed on a (1m) cube. This problem is included to illustrate the capabilities of the various approaches for large shearing deformations. Again, the material is Armco iron, material failure is not allowed, only half of the cube is computed as a plane of symmetry is used, and the boundaries have no restraints. Any failures are due to the numerical algorithm, not the material failure model. The initial velocities are shown in the upper left of Figure 15, and they impose an initial shear strain rate of 800 s^{-1}. The plastic deformations are essentially completed at t = 2000 μs. The two CPEM computations (particles only in lower left, and conversion of hexahedral elements into CPEM particles in lower right) provide results in good agreement with the hexahedral finite element results in the upper right. The GPA and Nodal EFG results with particles only do not provide accurate results. This same problem was run with tetrahedral elements and the CPEM results showed similar good agreement with the finite element result. The GPA and Nodal EFG results were better than for the hexahedral arrangement (particles only and conversion), but they were not nearly as accurate as the CPEM results. The reason for the large inaccuracies for the GPA and Nodal EFG results is not because these methods cannot represent shear deformations, but rather they cannot accurately represent the tension that occurs in the principal stress direction. If this same computation would be run under an applied hydrostatic pressure (such that no tensile stresses were generated) it would be expected that the results would be significantly improved.

3.5 Steel Projectile Impacts onto an Aluminum Target at 300 m/s

The fifth set of computations is shown in Figures 16 and 17, and it consists of a steel projectile impacting an aluminum target at an impact velocity of 300 m/s and an obliquity of 30 degrees (from normal). A distinguishing feature of this problem is that the projectile experiences minimal plastic strains, and that sliding between the projectile and the target is important. The tool steel projectile (EPIC library material # 10) [13] has a length of 8.89 cm and a diameter of 1.29 cm. The 1100 aluminum target (EPIC library material # 21) has a thickness of 2.63 cm.

The computed result in the upper left of Figure 16 is for a GPA target and a finite-element projectile composed of hedrahedral elements. All of the large deformations occur in the particle target, and the particles slide along the projectile elements. The residual velocity (V_r) is 204 m/s and the fraction of the initial kinetic energy (KE*) is 0.46. The computed result in the upper right is for a Nodal EFG target, with all other features identical to the GPA result in the upper left.

The result in the lower left is for a CPEM target and a hexahedral finite-element projectile. The sliding algorithm for this case is noted as "sliding-special" because there is only one converted material and this allows for use of the same sliding algorithm as used for the two previous computations. This sliding algorithm is more straightforward and accurate than is required for CPEM computations that have multiple converted materials interacting with one another [10].

With the exception of the particle-on-particle contact in the lower right, all of the results in Figure 16 are in good general agreement. The GPA target has the highest residual velocity (204 m/s) followed by the Nodal EFG (198 m/s) and CPEM (191 m/s). The reasons for these differences are not known. The GPA requires the least CPU time (581 min), followed by Nodal EFG (732 min) and CPEM (1332 min).

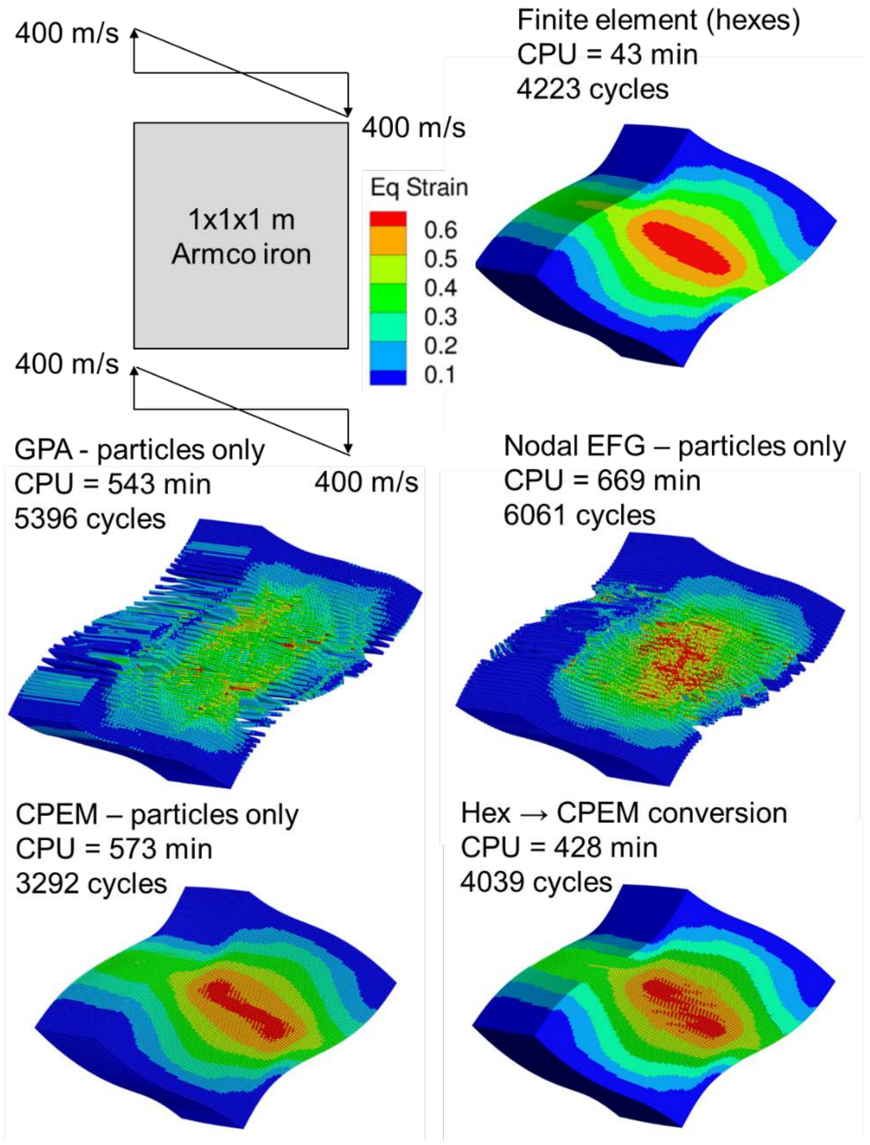

Figure 15. Shear deformation results for hexahedral finite element computation (top); GPA, Nodal EFG and CPEM algorithms (particles only); and conversion of hexahedral elements into CPEM particles (lower right). Final results (equivalent plastic strain) shown at t = 2000 μs.

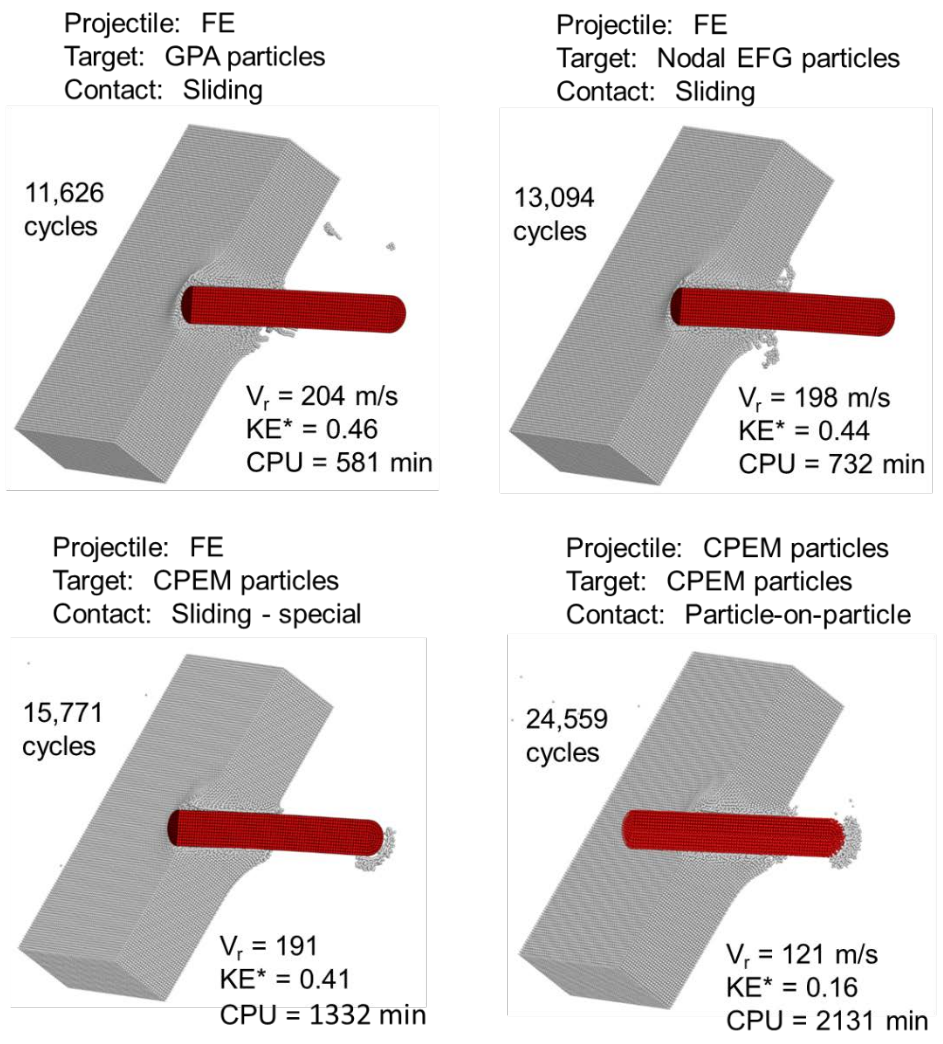

Figure 16. Computed results for a steel projectile impacting and perforating an aluminum target. Includes GPA, Nodal EFG and CPEM. Projectile is moving from left to right with an initial impact velocity of 300 m/s.

The results in Figure 17 are for conversion from hexahedral elements (left side) and from tetrahedral elements (right side). Again, these results are in good general agreement with those with particle-only targets (Figure 16), and the trends for residual velocity and run times are similar. The most significant difference between the conversion results in Figure 17 and the particle-only results in Figure 16 is that the conversion computations take much less computing time. Also, the conversions from tetrahedral elements (right side) generally provide slightly higher residual velocities. A possible explanation is that the computed pressures from the particles converted from tetrahedral elements are too low, and this leads to smaller failure strains generated by the Johnson-Cook failure model [15], which in turn leads to earlier failures and less target resistance.

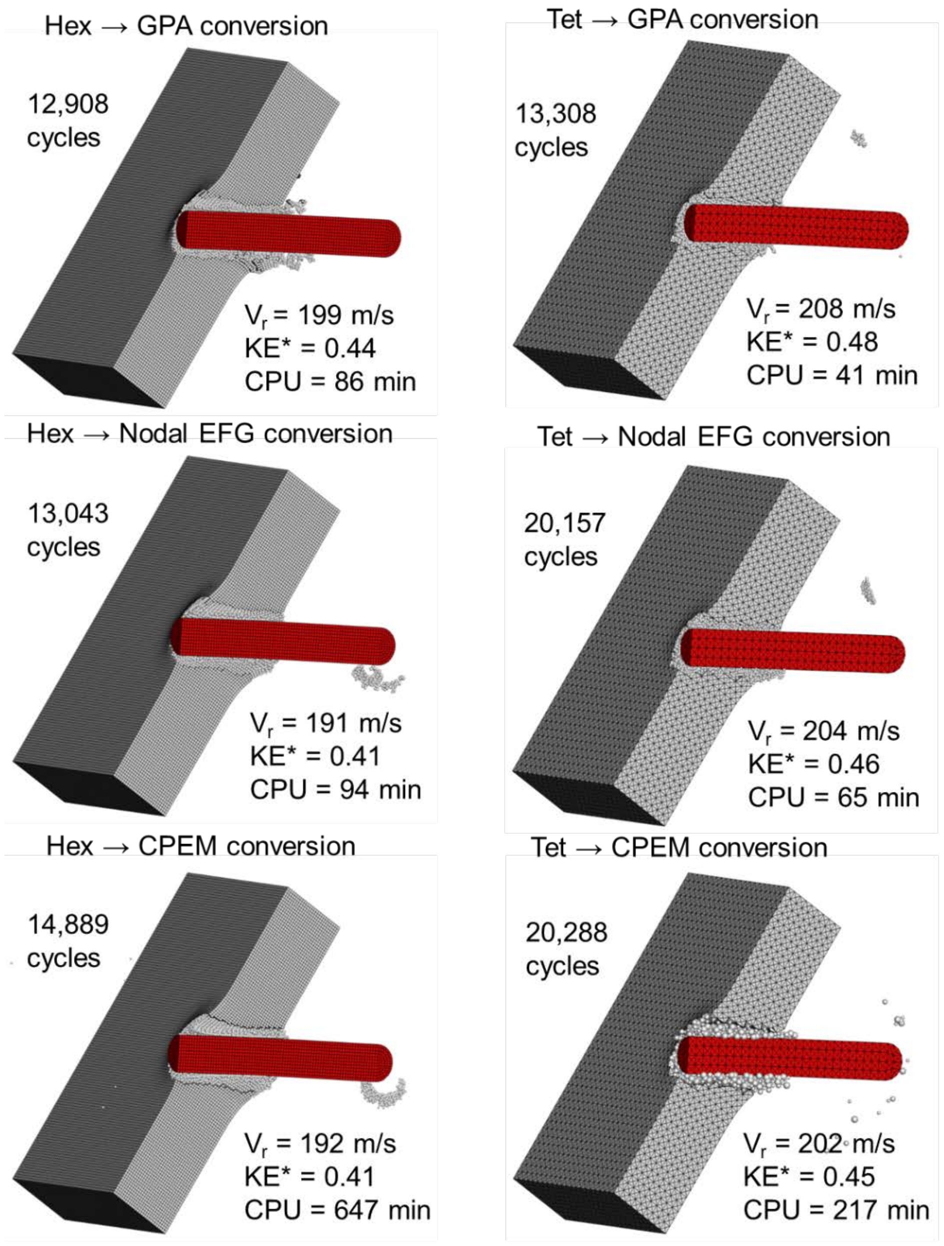

Figure 17. Computed results for a steel projectile impacting and perforating an aluminum target. Includes conversion of hexahedral and tetrahedral target elements into GPA, Nodal EFG and CPEM particles. Projectile is moving from left to right with an initial impact velocity of 300 m/s.

3.6 Tungsten Projectile Impacts onto a Steel Target at 1000 m/s

The sixth set of computations is shown in Figures 18 and 19, and it consists of a tungsten projectile impacting a steel target at an impact velocity of 1000 m/s with no obliquity (normal impact). This problem was selected because there are significant deformations in both the projectile and the target, and because fragments are formed from the backside of the target. The tungsten projectile (EPIC library material # 11) [13] has the same geometry as used for the previous problem in Figures 16 and 17, and the 4340 steel target (EPIC library material # 9) [13] also has the same geometry as the target in Figures 16 and 17. Some combinations of elements, particle algorithms, conversion or all particles, and sliding/contact algorithms did not run to completion. The GPA, with or without conversion, is clearly the most robust. All of the results include V_{tail}, which is the velocity of the centerline node on the aft end of the projectile. Damage distributions are provided, with red indicating damage above 0.95, where a damage of 1.0 indicates failure of the material [15].

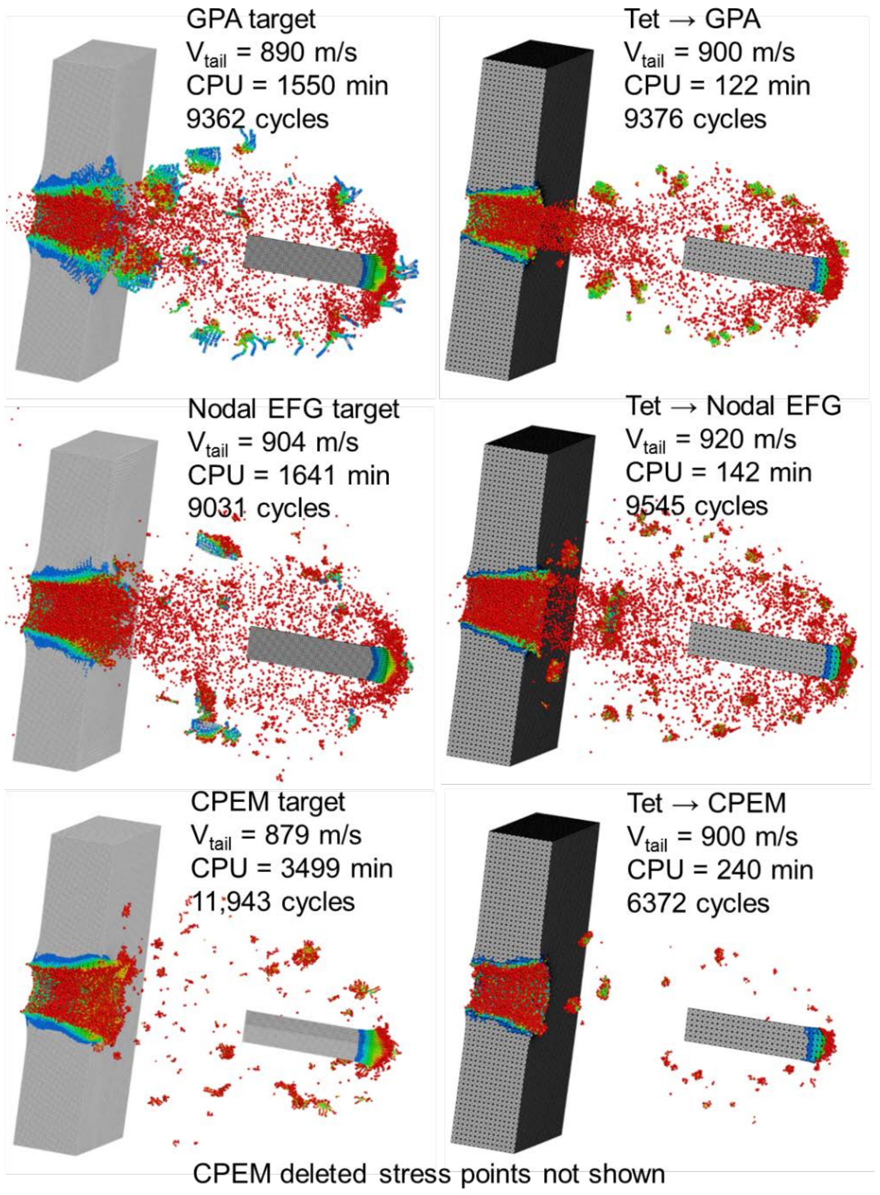

Figure 18. Computed results (including damage distributions) for a tungsten projectile impacting and perforating a steel target. Includes particles only and conversion from tetrahedral elements to GPA, Nodal EFG and CPEM particles. Projectile is moving from left to right with an initial impact velocity of 1000 m/s.

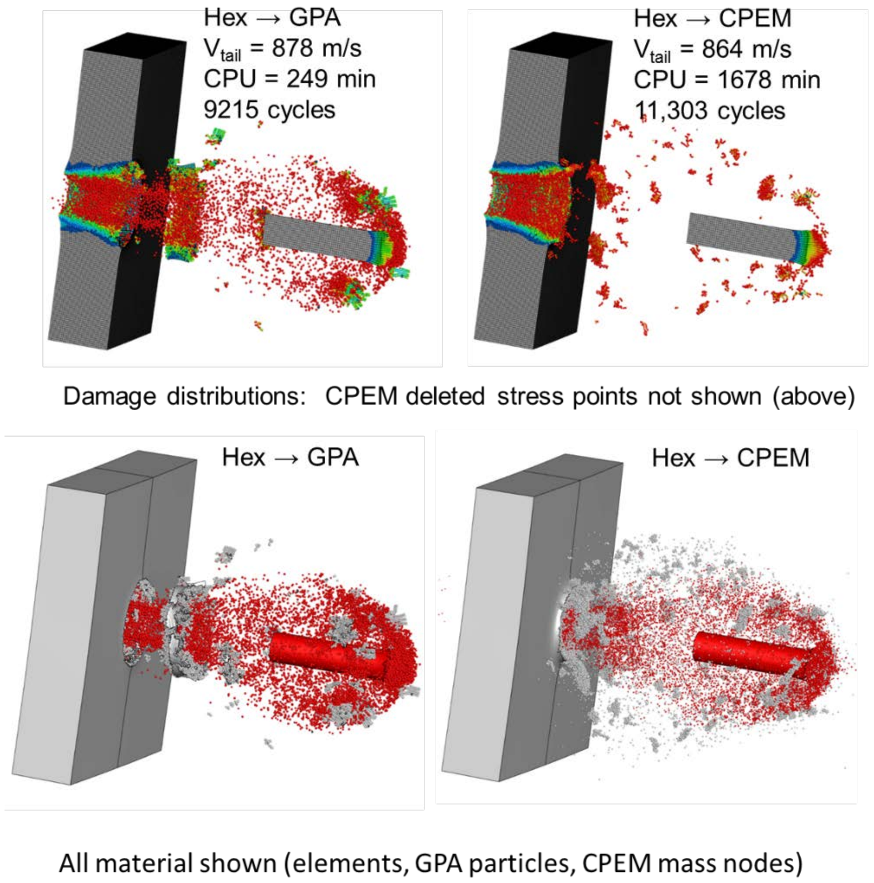

Figure 19. Computed results (including damage distributions) for a tungsten projectile impacting and perforating a steel target. Includes conversion from hexahedral elements to GPA (left) and to CPEM (right). Projectile is moving from left to right with an initial impact velocity of 1000 m/s.

The computations on the left side of Figure 18 are for the targets initially composed only of particles, and the computations on the right side are for tetrahedral elements converted into particles. For the particle-only targets the GPA and Nodal EFG computations allow the particles (target and converted elements from projectile) to slide along the finite-element projectile (initially composed of hexahedral elements). For the CPEM target (lower left) a particle-on-particle contact algorithm was used, and the velocity at the aft end of the projectile (V_{tail} = 878 m/s) is slightly less than the corresponding velocities for GPA and Nodal EFG. Recall that when the same particle-on-particle algorithm was used for the previous problem (lower right in Figure 16) the residual velocity was significantly reduced. This is because there is significant sliding for the lower-velocity problem, whereas the higher-velocity tends to form an open crater, and most of the interaction is between the forward end of the projectile and the target. Also, for CPEM the damage variable is carried at the stress points, and the stress points are deleted when they no longer have a good neighborhood. Therefore, it appears that there is less debris behind the target, but this is because the deleted stress points are not shown.

The right side shows results for initial tetrahedral meshes (projectile and target) that are converted to particles, and again the residual velocities are slightly higher than the particle-only targets on the left side. For the conversion to CPEM (lower right), the CPEM sliding/contact algorithm is used (which allows for conversion from two different materials) [10].

All of the results show the formation of fragments behind the target, but there are significant variations between the different computations. It is not possible to assess which approach is the most accurate as there are no test data for comparison. Even if there were, material model inaccuracies could affect the formation of the fragments. As a point of reference, some EPIC computations with GPA conversion were performed for a condition for which test data were available, and good general agreement was obtained [16].

The two additional computations in Figure 19 are for a hexahedral mesh converted into GPA particles (with sliding) and a hexahedral mesh converted into CPEM particles (with particle-on-particle contact). Damage distributions are shown on the top of the figure and materials are shown on the bottom. For the CPEM the stress points (representing damage) are not shown for deleted stress points (without good neighborhoods). For the materials on the bottom of the figure, all of the materials are shown. In addition, the entire problem is shown (both sides of the plane of symmetry). This provides a better visual representation of the fragment distributions. The fragments in this figure appear to be larger, and the residual velocities are slightly lower than those shown in Figure 18. These differences may or may not be due to the finer hexahedral mesh.

To summarize the results for this problem, the following observations can be made: Conversion computations require significantly less time than particle-only computations; the GPA is the most robust approach; particle-on-particle contact does not introduce significant inaccuracies; large fragments can be formed behind the target; and it is not clear which of the approaches is the most accurate.

4.0 IMPROVEMENTS IN PARALLELIZATION WITH OPEN MP

Broadly speaking, there have historically been two primary kinds of parallel computer architectures: shared memory, where all of the processors have access to all of the system memory; and distributed memory, where any processor will only have access to the memory associated with that processor. There have also been two major library tools to achieve parallelization in computational codes: Open Multi-Processing (OpenMP), and the Message Passing Interface (MPI).

MPI creates a framework that allows processes, which access different regions of memory in a distributed memory machine to pass messages and data to one another. The programmer is responsible for dividing the simulation into appropriately sized chunks for parallel execution, as well as maintaining all of the bookkeeping involved with insuring that the separate sections of the simulation correctly share their data.

OpenMP is called a lightweight parallelism—there is no need for the programmer to create a lot of bookkeeping or to subdivide the entire simulation. OpenMP allows the programmer to specify specific loops and sections of code that could be run in parallel, and the OpenMP library creates sub-processes called threads to handle pieces of the specified loops. Threads are not as substantial as a full process—they do not show up in the system's process manager. This means that they can do work with less overhead: they use less memory and system overhead than similar MPI worker processes.

Although OpenMP can run across distributed memory systems, it is most efficient on shared memory architectures. MPI can run on either architecture, but requires the initial programming investment, as well as the computational overhead and bookkeeping to keep track of the different pieces of the simulation.

At the time that EPIC was initially parallelized, it was difficult to achieve scaling past 16 processors using OpenMP, even on a shared memory architecture, and most of the supercomputers used by the DoD HPC community used distributed memory. Therefore, any attempts to make EPIC truly scalable required MPI, and so that was the direction that was chosen.

MPI computations continue to get faster as processors are added until the time spent in communication overwhelms the time spent doing the computation. Successive generations of distributed memory machines will utilize better hardware, reducing the communication times and increasing the scalability of programs without modification. MPI scaling will also run into issues of running out of memory: the amount of bookkeeping data will expand with the number of MPI processes. It should also be noted that load-balancing (making sure that each process is doing roughly the same amount of work as every other processor) is more difficult with MPI than with OpenMP.

The current HPC landscape is now very different, with tens of computational processors (called cores) being packed into a single hardware chip, and one or more chips will be part of a node, which shares memory. For instance, the ARL machine Centennial has 40 cores per node, where a node is a single section of discrete memory. It is of course possible to run MPI processes on each of those cores, but this adds to the bookkeeping data and system overhead for each of those MPI processes. Tonge [17] demonstrated that a hybrid OpenMP/MPI strategy works well: assigning one or two MPI processes per node, and then using OpenMP on the other cores of that node allows more parallelism while minimizing the bookkeeping overhead. The fact that

OpenMP threads require less memory than MPI processes also allows larger problems to be run in parallel on the same hardware.

The effort discussed herein started from Tonge's work, which had parallelized the element loop, the GPA particle loop, the node loop and triangular surface sliding with OpenMP compiler directives. Those directives were used as a template to parallelize other loops: the EFG particle loop, the CPEM particle loop, and the quadrilateral surface sliding.

Then a small scaling study was done on a conversion problem with an initial 153 million elements.

4.1 Expanding and Testing the OpenMP Implementation in EPIC

Testing parallel computations for correctness is challenging. The nature of parallel processing requires computations to be done in different locations and at different times, so summations of forces will frequently be done in a different order than the serial computation, leading to small round-off errors. For small problems running for small numbers of cycles, the parallel runs will be indistinguishable from the serial runs. For larger problems, especially ones that involve erosion or conversion, differences are inevitable. If the serial run computes an element's equivalent strain just below the erosion strain, and the parallel run computes that same element to be just above the erosion strain, then that cycle will mark a distinct difference between the serial and parallel runs in that location of the simulation. Those differences will propagate forward in time and outward from that location spatially, leading to more differences. For most simulations, these differences will not exceed 5% of the total energy, and the differences will frequently be less.

Years of experience with MPI parallelism have yielded some rules of thumb to determine whether the differences between parallel and serial runs are attributable to round-off, or whether there is a problem that needs to be investigated. EPIC generates a listing file that regularly records the simulation time, timestep, the total kinetic energy, and the total energy in the simulation. Noting when the first differences appear in the listing file, and the magnitude of the differences at the final time make reasonable tests for correctness.

Therefore, the differences between serial and OpenMP simulations will be compared to the differences between serial and MPI simulations to determine whether they are correct. For the first few simulations, data will be included from runs using the initial OpenMP effort by Tonge, which is available in the 2018 distribution of EPIC. The very first part of this effort was spent finding subtle race conditions in the initial OpenMP effort and fixing them. (Race conditions occur when different OpenMP threads access or modify a variable out of sync with each other. If e.g. processor A reads the variable first and then processor B modifies it, the solution will be correct. If processor B modifies the variable first, the solution will be wrong. Because it depends on which processor gets there first, it is called a race condition, and it will not occur consistently from run to run.) By comparing all three parallelization efforts (the original OpenMP implementation in the 2018 release, the current OpenMP implementation, and the current MPI implementation), it will hopefully be more clear what round-off errors look reasonable, and what errors should be examined more carefully.

Eight test conditions are considered. The first test is an axisymmetric cylinder impact problem using finite elements without erosion or sliding, in order to test the element loop (ELOOP) and nodal loop (NLOOP) compiler directives. This test uses the shortform to construct a medium triangular mesh of ARMCO Iron (Matl #4), to make a cylinder 2.54 cm long, 1.27 cm in diameter, impacting a rigid surface at 254 m/s, and the simulation lasts 50 μs. Because this is a cylinder impact problem, the final kinetic energy is very small compared to the total energy (in this case it is 0.01%). Therefore, even a large error in the final kinetic energy does not imply a bad solution; these errors are included simply to demonstrate the difference between a good solution and a better one. Table 1 compares the computed results for the axisymmetric cylinder impact problem. All error computations in these tables are treating the results of the serial run as the correct answer.

Table 1. Axisymmetric Cylinder Impact with Triangular Finite Elements

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Initial OpenMP	70/2854	0.32	83.97
Current OpenMP	2721/2854	0.0	0.85
MPI	2721/2854	0.0	1.07

The Initial OpenMP shows a difference much earlier in the computation (cycle 70 instead of cycle 2721). That, coupled with the worse final error in kinetic energy, indicates a probable race condition in the OpenMP directives, which was found and corrected for the Current OpenMP implementation.

The second test is a cylinder impact problem using GPA particles in order to test the particle loop (PLOOP) compiler directives. The results are provided in Table 2. This is using the same input as the previous problem, only requesting a medium mesh of GPA particles. For this serial run, the ratio of the final kinetic and total energies is 0.09%, and the same reasoning applies to including the final kinetic energy error. Again, the early first difference and larger final error points to a race condition in the Initial OpenMP implementation that was found and fixed for the Current OpenMP.

Table 2. Axisymmetric Cylinder Impact with GPA particles

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Initial OpenMP	20/2869	0.27	9.9
Current OpenMP	1559/2869	0.0	0.03
MPI	1499/2869	0.0	0.02

The third test involves 3D finite element erosion, in order to test the directives involved in 3D triangular surface sliding (SLIDE6). The results are provided in Table 3. This uses the 3D projectile-target shortform with an expanded coarse mesh of tetrahedral elements. A tungsten rod (EPIC material #11), with a length of 7.62 cm, diameter of 1.27 cm, is striking two spaced plates at a velocity of 1,524 m/s and an obliquity of 30 degrees. The first plate is 1.016 cm thick and made of 4340 Steel (EPIC material #9), there is a 2.032 cm gap and the second plate is also 1.016 cm thick, made of High Hard Armor (EPIC material #24). Unlike the first two cases, this final kinetic energy is over 95% of the total energy, and therefore errors in the final kinetic energy are more significant.

Table 3. 3D Erosion of Tetrahedral Finite Elements

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Initial OpenMP	900/2031	0.03	0.03
Current OpenMP	920/2031	0.02	0.03
MPI	920/2031	0.04	0.04

Note that based on the results in Table 3, there is no evidence of a race condition in the initial OpenMP run.

The fourth test involves the conversion of tetrahedral elements into GPA particles. It is identical to the third test except that the elements convert instead of erode, and they convert at a lower strain (30%).

Table 4. 3D Conversion of Tetrahedral Finite Elements into GPA particles

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Initial OpenMP	30/2509	0.06	0.09
Current OpenMP	1053/2509	0.01	0.00
MPI	1053/2509	0.00	0.04

The results for the fourth test are provided in Table 4, and show evidence of a race condition in the conversion case, even though it has minimal effect on the final energies of the problem. The race condition was in the tied slideline linking new GPA particles to the finite elements, and was corrected.

Having corrected the initial OpenMP effort, the directives from the initial effort were then used as a template to apply to other regions of the code. Subsequent tests will focus on the new OpenMP functionality, and therefore comparisons will only be between the Current OpenMP implementation and the current MPI implementation.

The fifth test is a cylinder impact problem using EFG particles in order to test the new EFGLOOP compiler directives, and the results are provided in Table 5. The full input file from the GPA test (test case 2) was modified to use EFG particles. The ratio of final kinetic energy to total energy is 0.09%, but the errors in the kinetic energy are still very small.

Table 5. Axisymmetric Cylinder Impact with EFG particles

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Current OpenMP	1769/2869	0.0	0.03
MPI	1499/2869	0.0	0.02

The sixth test is a cylinder impact problem using the CPEM in order to test the directives in the CPEM loop (EPLOOP), and the results are documented in Table 6. This run took the full input file from the triangular element cylinder impact and turned it into a CPEM run with a conversion strain of 0.3. The ratio of final kinetic energy to total energy is 0.06%.

Table 6. Axisymmetric Cylinder Impact with the CPEM

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Current OpenMP	649/978	0.0	0.03
MPI	549/978	0.0	0.02

The seventh test (Table 7) involves the erosion of hexahedral elements to test the 3D quadrilateral surface sliding (SLIDE6Q). The full input for the tetrahedral erosion problem was modified such that every symmetric brick was turned into a hex element. Therefore, this hex mesh is substantially coarser than the tetrahedral mesh. In this case, both the OpenMP and the MPI computations match the serial computation perfectly (at least to the six significant digits reported in the listing file).

Table 7. 3D Erosion of Hexahedral Finite Elements

	Cycle Number of First Difference/Total Serial Cycles	Final Error in Total Energy (%)	Final Error in Kinetic Energy (%)
Current OpenMP	N/A	0.0	0.0
MPI	N/A	0.0	0.0

The eighth and final test (Table 8) involves the conversion of hexahedral elements. It takes the input from the erosion of hexahedral elements and turns it into a conversion problem with a conversion strain of 30%. In this case, the MPI computation matches the serial computation perfectly, but the OpenMP does not.

Table 8. 3D Conversion of Hexahedral Finite Elements into GPA Particles

	Cycle Number of First Difference/Total Serial Cycles	**Final Error in Total Energy (%)**	**Final Error in Kinetic Energy (%)**
Current OpenMP	80/660	0.12	0.82
MPI	N/A	0.0	0.0

Even though the final errors of the Current OpenMP implementation are small, they are larger than the MPI errors. In addition, the first difference happens early in the computation at 80 cycles, so the current OpenMP implementation appears to have a subtle race condition somewhere in the hexahedral element conversion code, but it has not been located yet.

4.2 Scaling Test with the OpenMP/MPI Hybrid Code on a Large Problem (153 million elements)

The ultimate goal of this work is to get EPIC to run problems involving hundreds of millions of elements in reasonable amounts of time by being able to scale up to thousands of cores. Tonge has already shown that the OpenMP/MPI hybrid method is capable of scalability on a larger number of cores than either MPI or OpenMP would be capable of delivering on their own. The problem is discovering the barriers and bottlenecks that prevent better scalability, and determining approaches to overcome them.

The original intent of this project was to quickly copy the OpenMP directives into other sections of the EPIC code and spend the majority of the time investigating the barriers and bottlenecks to massive scalability. Unfortunately, debugging the new OpenMP directives proved to be more challenging and time consuming than expected, and so what was anticipated to be a small effort turned into a major effort reducing the time spent on scaling. Although reduced in scope, the scaling study did provide some helpful results as discussed next.

The chosen problem was a Tool Steel rod (EPIC material #10) with a length of 0.08 m and a diameter of 0.04 m impacting a stacked plate target. The top three plates are Silicon Carbide (SiC-B, EPIC material #166) and are 0.02 m thick, and the fourth plate is 6061 Al (EPIC material #23) and is 0.06 m thick. All four plates are 1.6 m x 1.6 m in extent. By varying the number of rings in the rod and the corresponding refinement of the target mesh, input files were generated that would create anywhere from 5 million to 687 million elements.

The computations completed for this scaling study used 6 element rings in the projectile rod, with 153 million elements. Because rapid turnaround was desired for these runs, they were limited to 2 hours of wall clock time, and 10 μs of simulation time.

All of the reported scaling runs were done on the SGI ICE XA machine named Centennial at the Army Research Laboratory (ARL). Centennial has 40 cores per node. The first runs were done with only one MPI process per node, and 40 OpenMP threads. EPIC has a parallel pre-processor, but it is currently only MPI parallel, so a two node run means that 153 million elements are being pre-processed by only two MPI processes, and that takes well over half of the two-hour allowed run time. Once the preprocessing has been done once, however, starting from the initial restart is much faster than the initial pre-processing. However, later it will be shown that the start-up time from the restart file is not scaling.

After several establishing runs with one MPI process per node, the number of MPI processes per node was varied on runs using 16 nodes. Using 4 MPI processes (with 10 OpenMP threads) per node was more efficient than either 2 or 8 MPI processes per node.

The results of those runs using 4 MPI processes per node are presented in Figure 20.

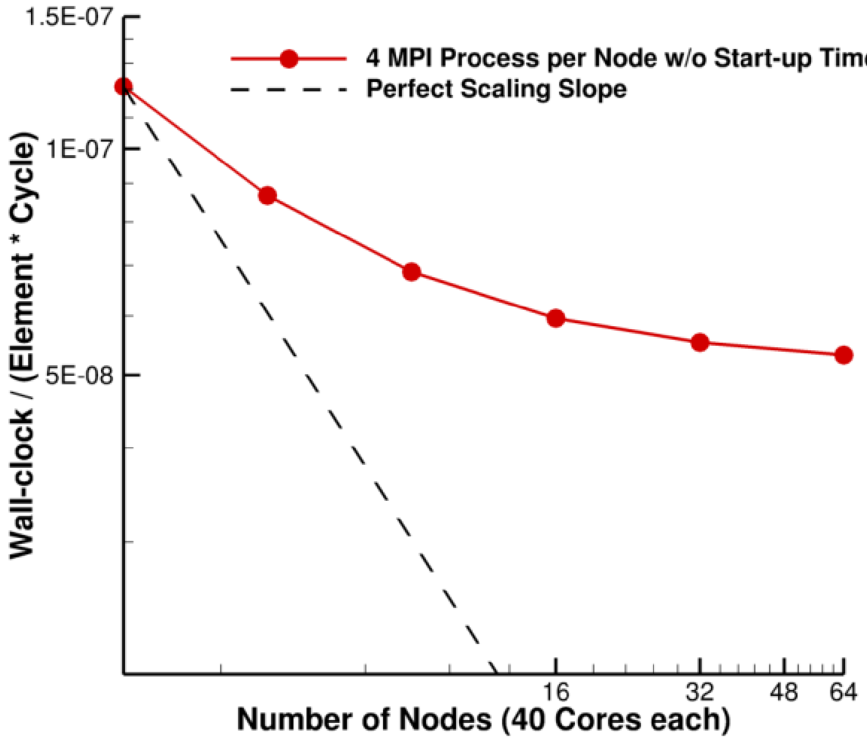

Figure 20. Scaling for a 153 million element problem on Centennial.

The start-up times are not included in the scaling computation because those times are not contributing to the progression of the simulation. Longer simulations would amortize those start-up times more effectively. Although the problem is still scaling out to 64 nodes (2,560 cores), the gain from 32 nodes to 64 nodes is only about 4%. Because the problem is stopped after 10 microseconds, the number of particles in the problem is still small, and the sliding costs of those particles are being dwarfed by the rest of the inter-plate sliding. Runs modeling the entire impact event may well end up with a different scaling curve.

The raw timing data from the first four of those runs are presented in Table 9. The subdivided times come from the average of all the MPI processes, so the sub-categories will not exactly sum to the total time.

Table 9. Raw Timing Data from Several Runs on Centennial (seconds)

	2 Nodes	**4 Nodes**	**8 Nodes**	**16 Nodes**
Element Loop	2840	1453	762	418
Nodal Loop (w/o sliding)	255	264	248	225
Sliding	490	394	358	323
Particle Loop	165	154	166	192
Communication	1282	1336	1317	1313
Start-up costs	203	305	514	936
Total Time	5259	3927	3381	3422

The total time goes up when moving from 8 to 16 nodes, because the start-up time associated with the distribution of the initial mesh is unfortunately proportional to the number of processors, even though it should not be. (Again, if the simulation were run longer, the start-up cost would be a smaller fraction of the total, and the total time would more accurately reflect the scaling of the simulation computations.)

From the results in Table 9, it can be seen that the element loop is scaling reasonably well. The nodal loop (without sliding) is not showing substantial scaling, and should be examined. The sliding loop is scaling, but perhaps not as well as it could be. Tonge suggested that the subroutine BUCK3 might be the main sticking point there, but there was not enough time to try to get that to work. The particle loop is not scaling well, but the small number of particles in the computation contribute to that.

While the start-up time is substantially less if using a restart instead of starting from the parallel pre-processor, users prefer to be able to submit a single run instead of running the pre-processor separately. Also, the sooner the code reaches the actual computation, the sooner any problems in the setup of the simulation will be discovered. For these reasons, the time required by the parallel pre-processor and the distribution of the initial restart should be addressed. Even if they technically do not affect the scaling of the simulation computations, long setup times impact the user's experience negatively.

5.0 CONCLUSIONS

For the meshless particle study, the following conclusions can be made:

- Meshless-particle approaches can be very useful in performing Lagrangian computations for problems that involve severe distortions.
- Meshless particle approaches generally take significantly more computer time than finite-element approaches.
- Conversion from distorted finite elements into meshless particles can be a powerful approach. It allows for accurate finite-element responses for the lower-strained portions of the problem, accurate sliding between finite-element surfaces that are not severely distorted, run times that are significantly reduced when compared to computations that use particles only, and robust solutions for the highly distorted regions of the problem.
- The GPA is the most robust, the least accurate, and runs the fastest.
- The Nodal EFG approach is less robust than the GPA, more accurate than GPA and less accurate than CPEM. It runs slower than the GPA and faster than the CPEM.
- The CPEM is generally the most accurate and requires the most run time. It is less robust than GPA and about the same as Nodal EFG. When multiple materials are converted and interact with one another, the sliding/contact algorithm can lead to decreased robustness and increased run times.
- Conversion from hexahedral elements is sometimes more accurate than conversion from tetrahedral elements. This appears to be most evident with the pressure computations. The inaccuracies of the pressure may be a greater issue for materials whose strength (and failure) is dependent on the pressure (such as ceramics and concrete). Generally, tetrahedral elements are more robust under large strains, but that may not be an issue when the elements are converted at moderate strains.
- For some of the comparative problems, (wave propagation, expanding spherical shell and shear deformations) the GPA and Nodal EFG approaches did not provide accurate results. For the ballistics problems, however, the GPA and Nodal EFG approaches provided results that appeared to be comparable to those obtained by CPEM.
- It is possible to compute the formation of fragments from the backside of a target perforated by a projectile. This can be accomplished with all three meshless-particle algorithms (GPA, Nodal EFG, CPEM), with or without conversion. A wide range of results is obtained with the various approaches, however, and it is not clear which approaches are the most accurate.

For the parallelization improvement study, the following conclusions can be made:

- As already demonstrated by Tonge, the OpenMP/MPI hybrid method delivers scalability with EPIC for problems involving hundreds of millions of elements using thousands of cores.

- The OpenMP compiler directives have now been applied to the EFG and CPEM particle algorithms, as well as to the quadrilateral sliding algorithm, expanding the scope of problems that can be run with massive scalability. They have also been verified for correctness by comparison with serial results. There still remains an issue with the OpenMP implementation when converting hexahedral elements to particles.
- While not directly impacting the scalability of the computations, the times required by the parallel pre-processor and the distribution of the initial restart data impact the total computational time required, and therefore the user's experience and they should be remedied to the extent possible.
- Further work should include the following:
    - Identify and fix the race condition for hexahedral conversion using OpenMP.
    - Evaluate and if possible improve the scaling in the Nodal Loop, Sliding Loop, Particle Loop, and BUCK3 subroutine.
    - Evaluate and if possible improve the start-up times from a restart file.
    - Evaluate and if possible add OpenMP directives to the parallel pre-processor.

6.0 REFERENCES

1. Johnson GR. Numerical algorithms and material models for high-velocity impact computations. Int J Impact Eng **38**, 2011.
2. Johnson GR, Stryk RA, Dodd JG. Dynamic Lagrangian computations for solids, with variable nodal connectivity for severe distortions. Int J Numer Methods Eng **23**, 1986.
3. Libersky LD, Petschek AG. Smooth particle hydrodynamics with strength of materials. Lecture notes in physics 395. In: HE Trease, MJ Fritts, WP Crowley, editors. Advances in the Free-Lagrange Method, Berlin Heidelberg Germany: Springer-Verlag, 1991.
4. Johnson GR, Peterson EH, Stryk RA. Incorporation of an SPH option into the EPIC code for a wide range of high velocity impact computations. Int J Impact Eng **14**, 1993.
5. Johnson GR, Stryk RA, Beissel SR. SPH for high velocity impact computations. Comp Meth Appl Mech Eng **139**, 1996.
6. Johnson GR, Beissel SR, Stryk RA. A generalized particle algorithm for high velocity impact computations. Comput Mech **25**, 2000.
7. Johnson GR, Stryk RA. Conversion of 3D distorted elements into meshless particles during dynamic deformation. Int J Impact Eng **28**, 2003.
8. Johnson GR, Stryk RA. Symmetric contact and sliding surface algorithms for intense impulsive loading computations. Comp Meth Appl Mech Eng **190**, 2001.
9. Johnson GR, Beissel SR, Gerlach CA. A 3D combined particle-element method for intense impulsive loading computations involving severe distortions. Int J Impact Eng **84**, 2015.
10. Gerlach CA, Johnson GR. A contact and sliding interface algorithm for the combined particle-element method. Int J Impact Eng **113**, 2018.
11. Belytschko T, Lu YY, Gu L. Element-free Galerkin methods. Int J Numer Methods Eng **37,** 1994.
12. Beissel S, Belytschko T. Nodal integration of the element-free Galerkin method. Comp Meth Appl Mech Eng **139**, 1996.
13. Johnson GR, Cook WH. A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures, In: Proceedings of the Seventh International Symposium on Ballistics. The Hague, the Netherlands. 1983.
14. Swegle JW, Attaway SW, Heinstein FJ, Mello FJ, Hicks DL. An analysis of smoothed particle hydrodynamics. SAND93-2513, Sandia National Laboratories, Albuquerque, MN 1994.
15. Johnson GR, Cook WH. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. Eng Fract Mech **21**, 1985.
16. Johnson GR, Stryk RA, Gerlach CA, Holmquist TJ, Rowe NL. A quantitative assessment of computational results for behind armor debris. In: Francisco Galvez, Vicente Sanchez- Galvez, editors. In: Proceedings of 23rd International Symposium on Ballistics. Tarragona: Graficas Couche, Madrid Spain, 2007.
17. Tonge A. Initial Feasibility Study for Using a Hybrid MPI/OpenMP Approach in Elastic Plastic Impact Computation (EPIC). Report ARL-MR-0979. Weapons and Materials Research Directorate, ARL. Aberdeen MD, 2018.