INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng. 2000; **47**:1323–1341

Parallel computation of meshless methods for explicit dynamic analysis

Kent T. Danielson[1,2,*,†], Su Hao[1], Wing Kam Liu[1], R. Aziz Uras[3] and Shoafan Li[1]

[1]*Department of Mechanical Engineering and Army High Performance Computing Research Center, Northwestern University, 2145 Sheridan Road, Evanston, IL 60208-3111, U.S.A.*
[2]*U.S. Army Engineer Research and Development Center, Waterways Experiment Station, 3909 Halls Ferry Road CEERD-SD-R, Vicksburg, MS 39180-6199, U.S.A.*
[3]*Reactor Engineering Division, Argonne National Laboratory, 9700 South Cass Avenue, Building 208, Argonne, IL 60439-4842, U.S.A.*

SUMMARY

A parallel computational implementation of modern meshless methods is presented for explicit dynamic analysis. The procedures are demonstrated by application of the Reproducing Kernel Particle Method (RKPM). Aspects of a coarse grain parallel paradigm are detailed for a Lagrangian formulation using model partitioning. Integration points are uniquely defined on separate processors and particle definitions are duplicated, as necessary, so that all support particles for each point are defined locally on the corresponding processor. Several partitioning schemes are considered and a reduced graph-based procedure is presented. Partitioning issues are discussed and procedures to accommodate essential boundary conditions in parallel are presented. Explicit MPI message passing statements are used for all communications among partitions on different processors. The effectiveness of the procedure is demonstrated by highly deformable inelastic example problems. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: parallel computing; meshless methods; reproducing kernel particle methods; large deformation; inelasticity; explicit dynamics.

1. INTRODUCTION

A variety of new meshless modelling method have recently emerged. Particle [1], Element Free Galerkin [2] (EFG), cloud [3], and other kernel [4–6] and partition of unity [7] methods are

*Correspondence to: Kent T. Danielson, U.S. Army Engineer Research and Development Center, Waterways Experiment Station, 3909 Halls Ferry Road, CEERD-SD-R, Vicksburg, MS 39180-6199, U.S.A.
†E-mail: danielk@wes.army.mil

Contract/grant sponsor: Army High Performance Computing Research Center; contract/grant number: DAAH04-95-2-003/DAA-95-C-0008
Contract/grant sponsor: U.S. Department of Energy, Technology Support program; contract/grant number: W-31-109-Eng-38

CCC 0029-5981/2000/071323–19$17.50

Received 11 January 1999
Revised 18 May 1999

examples that compete, for various applications, with finite elements, boundary elements, and classical meshless methods (e.g. Rayleigh–Ritz). These methods may be attractive for certain cases, since they possess various characteristics that overcome some of the shortcomings of more traditional methods. Large parallel computings systems have also become commonly available. Effective utilization of such systems generally involves explicit message passing statements in conjunction with elaborate load balancing-communication minimizing schemes. Parallel computing is especially attractive for modern meshless methods, since they frequently require more computations than other competitive techniques. In addition, the applications where the new meshless methods can be advantageous are usually highly complex and thus computationally intensive.

The present treatment will focus on the Reproducing Kernel Particle Method [4–6, 8] (RKPM) for explicit dynamic analysis of solid and structures, but the procedures are directly applicable to some other meshless methods (e.g. EFG). Several distinct advantages of RKPM are its ability to accurately model extremely large deformations without mesh distortion problems and its ease to adaptive modelling by simply changing particle definitions for desired refinement regions. An overview of a Lagrangian RKPM for non-linear explicit dynamic analysis is first given and a new method for enforcement of essential boundary conditions is presented. A general description of the parallel implementation is then described. The parallel procedure primarily consists of a mesh partitioning pre-analysis phase, a parallel analysis phase that includes explicit message passing among partitions on separate processors, and a post-analysis phase for gathering separate processor output files into a single coherent data structure for post-processing. Load distribution and message passing minimization aspects are outlined, and data structures for processing on large scalable computing platforms are described. Benefits of the essential boundary condition enforcement method are identified for parallel processing and application to large strain/deformation inelastic example problems is provided.

2. REPRODUCING KERNEL PARTICLE METHOD

For spatial position, $\mathbf{x}$, RKPM formulations use a kernel approximation to the displacements, $\mathbf{u}(\mathbf{x})$, such that

$$\mathbf{u}^K(\mathbf{x}) = \int_{-\infty}^{\infty} \mathbf{u}(\tilde{\mathbf{x}}) K(\mathbf{x} - \tilde{\mathbf{x}})\, \mathrm{d}V \tag{1}$$

where $K(\mathbf{x} - \tilde{\mathbf{x}})$ is a kernel function which is typically taken as the SPH particle interpolation function that is modified by a correction function to accommodate the presence of boundaries of finite domains in order to satisfy consistency (reproducing) conditions. Evaluation of Equation (1) produces an interpolation in terms of NP particle values, $\mathbf{d}_J$, within the subregion of influence (see Figure 1)

$$\mathbf{u}^K(\mathbf{x}) = \sum_{J=1}^{\mathrm{NP}} N_J(\mathbf{x}) \mathbf{d}_J = \mathbf{N}^{\mathrm{T}}(\mathbf{x}) \cdot \mathbf{d}^S \tag{2}$$

where $\mathbf{d}^S$ is the vector of generalized particle displacements in the support zone and $\mathbf{N}(\mathbf{x})$ is a matrix of RKPM interpolation functions (sometimes referred to as shape functions). The size and shape of the influence region can be general and can vary within the domain and between different analyses. The support nodes are determined by searching within each individual support

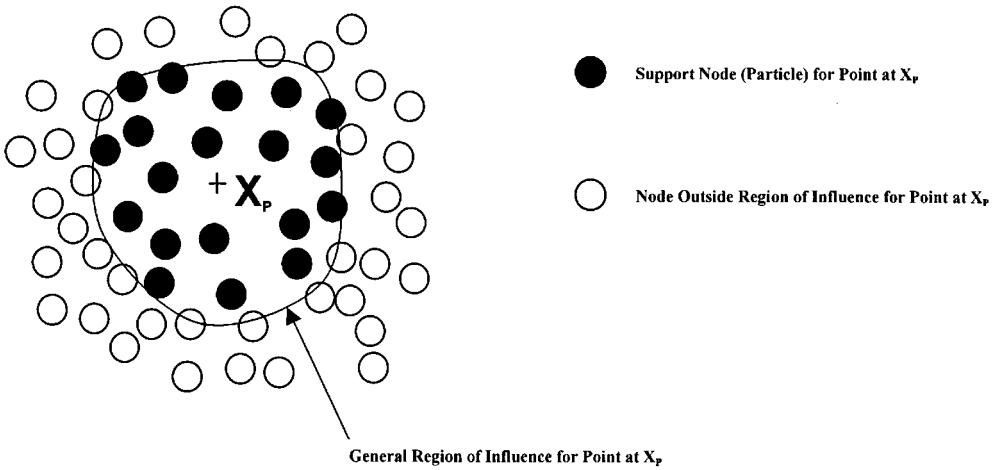

Figure 1. Support nodes for RKPM interpolation.

zone for any desired point (e.g. node, integration point, output location, etc.). Detailed descriptions of RKPM procedures can be found, for example, in References [4–6, 8].

A distinct characteristic of the interpolation in Equation (2) is that it is non-local (i.e. the Kronecker delta property is not satisfied). The displacements at particles are not the particle values, but are determined by the values of all particles in its proximity. This is

$$\mathbf{u}^K(\mathbf{x}_I) = \sum_{J=1}^{NP} N_J(\mathbf{x}_I)\mathbf{d}_J \neq \mathbf{d}_I \tag{3}$$

Particles (frequently referred to as nodes) are distributed with regard to the degree of interpolation refinement. To evaluate the governing particle equations (see next section), many more integration points than nodes are frequently required. A typical RKPM model is shown in Figure 2.

3. EXPLICIT DYNAMIC ANALYSIS

3.1. Lagrangian form of the governing equations

Using virtual work [9], the large strain/deformation equations for dynamic equilibrium at any time, t, are

$$\int_{V_d} \rho_d \ddot{\mathbf{u}} \cdot \delta\mathbf{u}\, dV_d = \int_{V_d} \rho_d \mathbf{b} \cdot \delta\mathbf{u}\, dV_d + \int_{A_d} \hat{\mathbf{t}} \cdot \delta\mathbf{u}\, dA_d - \int_{V_d} \boldsymbol{\sigma} : \frac{\partial \delta\mathbf{u}}{\partial \mathbf{x}}\, dV_d \tag{4}$$

where δ is the variational operator, each dot refers to differentiation with respect to time, ρ_d is the current mass density, $\mathbf{b}$ are body forces, $\hat{\mathbf{t}}$ are applied surface tractions, $\boldsymbol{\sigma}$ are cauchy stresses, and A_d and V_d are the deformed configuration surface area and volume, respectively. The equivalent

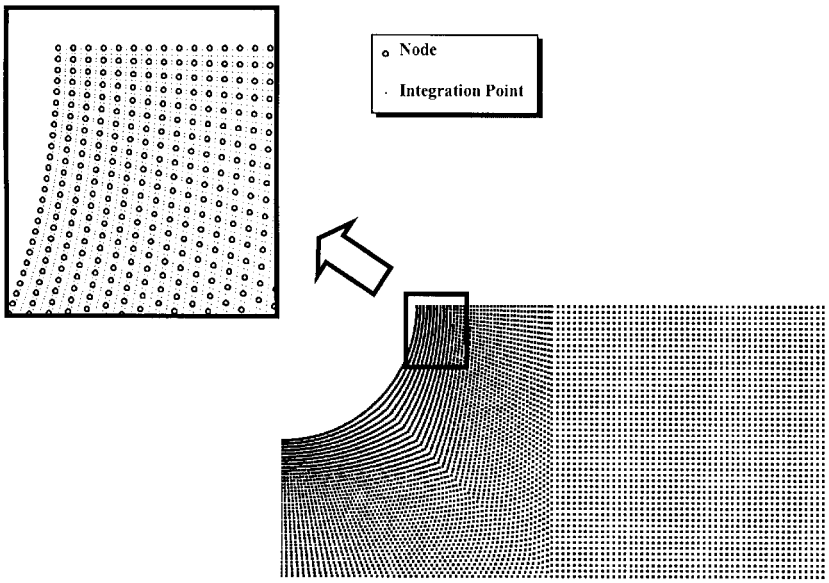

Figure 2. RKPM model of a plate with a circular cutout (6329 particles, 55 296 integration points).

virtual power statement is obtained by differentiating the virtual displacements with respect to time. The current position, $\mathbf{x}$, is related to the undeformed position, $\mathbf{X}$, by

$$\mathbf{x} = \mathbf{X} + \mathbf{u}(\mathbf{X}, t) \tag{5}$$

A Lagrangian formulation is chosen so that all quantities are referred to the undeformed configuration. Thus

$$\mathbf{u}^K(\mathbf{X}, t) = \sum_{J=1}^{\mathrm{NP}} N_J(\mathbf{X})\mathbf{d}_J(t) = \mathbf{N}^{\mathrm{T}}(\mathbf{X}) \cdot \mathbf{d}^S(t) \tag{6a}$$

$$\frac{\partial \mathbf{u}^K(\mathbf{X}, t)}{\partial \mathbf{X}} = \sum_{J=1}^{\mathrm{NP}} \frac{\partial N_J(\mathbf{X})}{\partial \mathbf{X}}\mathbf{d}_J(t) = \frac{\partial \mathbf{N}^{\mathrm{T}}(\mathbf{X})}{\partial \mathbf{X}} \cdot \mathbf{d}^S(t) \tag{6b}$$

The Lagrangian formulation provides tremendous computational benefits, since the search for neighbouring support nodes and the interpolation function calculations in Equation (6) are only done once at the beginning of the analysis. As shown in following sections, the Lagrangian formulation is also advantageous for the present essential boundary condition enforcement and parallel processing approaches.

The interpolation of Equation (6) is inserted into Equation (4), so that the resulting approximation is equivalent to a Galerkin weak form of the momentum equation (equations of motion). To simplify the use of Equation (6) with the virtual work statement, the following properties of the deformation gradient, $\mathbf{F}$, and conservation of mass are used to define an alternate form of Equation (4)

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \mathbf{I} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} \tag{7}$$

$$|\mathbf{F}|\,\mathrm{d}V_0 = \mathrm{d}V_\mathrm{d} \tag{8}$$

$$\rho_0 \mathrm{d}V_0 = \rho_\mathrm{d}\,\mathrm{d}V_\mathrm{d} \tag{9}$$

so that

$$\int_{V_0} \rho_0\,\ddot{\mathbf{u}}\cdot\delta\mathbf{u}\,\mathrm{d}V_0 = \int_{V_0} \rho_0\,\mathbf{b}\cdot\delta\mathbf{u}\,\mathrm{d}V_0 + \int_{A_\mathrm{d}} \hat{\mathbf{t}}\cdot\delta\mathbf{u}\,\mathrm{d}A_\mathrm{d} - \int_{V_0} \boldsymbol{\sigma}\colon\frac{\partial\delta\mathbf{u}}{\partial\mathbf{x}}\,|\mathbf{F}|\,\mathrm{d}V_0 \tag{10}$$

where the subscript, 0, refers to the undeformed configuration. Second Piola–Kirchhoff stresses, **S**, and Green–Lagrange strains, **E**, may also be used to refer integrand quantities within the internal virtual work expression to the undeformed configuration.

$$\int_{V_0} \mathbf{S}\colon\delta\mathbf{E}\,\mathrm{d}V_0 = \int_{V_\mathrm{d}} \boldsymbol{\sigma}\colon\frac{\partial\delta\mathbf{u}}{\partial\mathbf{x}}\,\mathrm{d}V_\mathrm{d} = \int_{V_0} \boldsymbol{\sigma}\colon\frac{\partial\delta\mathbf{u}}{\partial\mathbf{x}}\,|\mathbf{F}|\,\mathrm{d}V_0 \tag{11}$$

This form is especially effective for elastic and hyperelastic material models,, whereby **S** is easily computed. For other constitutive models, $\boldsymbol{\sigma}$ is more convenient so that either Equation (10) is evaluated directly, or Equation (11) is used in conjunction with the following relationship:

$$\mathbf{S} = \frac{\rho_0}{\rho_\mathrm{d}}\,\mathbf{F}^{-1}\cdot\boldsymbol{\sigma}\cdot\mathbf{F}^{-\mathrm{T}} = |\mathbf{F}|\,\mathbf{F}^{-1}\cdot\boldsymbol{\sigma}\cdot\mathbf{F}^{-\mathrm{T}} \tag{12}$$

Finally, derivatives with respect to the deformed configuration can be conveniently obtained by

$$\frac{\partial(\ )}{\partial\mathbf{x}} = \mathbf{F}^{-1}\cdot\frac{\partial(\ )}{\partial\mathbf{X}} \tag{13}$$

An advantage of the forgoing Lagrangian RKPM formulation is that it may accurately perform highly deformable analyses without mesh distortion problems. RKPM does not use elements that may become poorly shaped during such analyses. Instead, the formulation accuracy is governed by the ability to invert the deformation gradient tensor, which is assured when the determinant of **F** is greater than zero.

3.2. Enforcement of essential boundary conditions

For an approximation with the virtual work principle, the Essential Boundary Conditions (EBC) must be satisfied directly by the interpolation functions or accommodated by augmenting the variational statement with constraints. A major difference between RKPM and other methods (e.g. finite element) is the manner in which essential boundary conditions can be enforced. The non-local interpolation condition of Equation (3) poses an additional computational challenge. Whereas essential boundary conditions for finite elements are imposed locally at nodes (because they possess the Kronecker delta property), EBC enforcement with RKPM is non-local over a patch of particles/nodes. In some cases, the EBCs can be adequately approximated by local specification at the nodes (assuming a Kronecker delta property). This approximation can be accurate, by St. Venant's Principle, when the primary regions of interest are away from the EBCs. In general, however, a coupled set of equations is usually solved, even for explicit analyses. Previous efforts used Lagrange multipliers to constrain the variational statement [2] or a set of simultaneous equations was directly solved [8]. In either case, significant computations were

generally necessary to enforce the essential boundary conditions. These procedures also are not well suited for parallel processing, since they must generally be made over multiple processors.

The authors now propose an alternate approach that may also require significant computational effort, as it is algebraically equivalent to other existing equation solving methods. By treating the imposition as a transformation of the interpolation functions, however, this form is better for parallel processing (discussed in Section 4). Describing the EBC equations by

$$g_{\mathrm{B}} = u^K(\mathbf{X}_{\mathrm{B}}) = \sum_{J=1}^{\mathrm{NP}} N_J(\mathbf{X}_{\mathrm{B}})\mathbf{d}_J \tag{14}$$

where g_{B} is the specified EBC at $\mathbf{X}_{\mathrm{B}}$ or can be an integral relation along $\mathbf{X}_{\mathrm{B}}$ (e.g., zero displacement along a edge). In the matrix form for all conditions

$$\mathbf{G}^{\mathrm{T}} \cdot \mathbf{d} = \mathbf{g} \tag{15}$$

whereby Gram–Schmidt or Householder orthogonalization

$$\mathbf{G}^{\mathrm{T}} \cdot \mathbf{G} = \mathbf{I} \tag{16}$$

$$\mathbf{J}^{\mathrm{T}} \cdot \mathbf{G} = \mathbf{0} \tag{17}$$

$$\mathbf{J}^{\mathrm{T}} \cdot \mathbf{J} = \mathbf{I} \tag{18}$$

The generalized variables are now represented by

$$\mathbf{d} = \mathbf{J} \cdot \tilde{\mathbf{d}} + \mathbf{G} \cdot \mathbf{g} \tag{19}$$

Equation (19) inserted into Equation (6) produces

$$\mathbf{u}^K(\mathbf{X}) = \mathbf{N}^{\mathrm{T}}(\mathbf{X}) \cdot \mathbf{J} \cdot \tilde{\mathbf{d}} + \mathbf{N}^{\mathrm{T}}(\mathbf{X}) \cdot \mathbf{G} \cdot \mathbf{g} \tag{20a}$$

$$\frac{\partial \mathbf{u}^K(\mathbf{X})}{\partial \mathbf{X}} = \frac{\partial \mathbf{N}^{\mathrm{T}}(\mathbf{X})}{\partial \mathbf{X}} \cdot \mathbf{J} \cdot \tilde{\mathbf{d}} + \frac{\partial \mathbf{N}^{\mathrm{T}}(\mathbf{X})}{\partial \mathbf{X}} \cdot \mathbf{G} \cdot \mathbf{g} \tag{20b}$$

or

$$\mathbf{u}^K(\mathbf{X}) = (\mathbf{N}^J(\mathbf{X}))^{\mathrm{T}} \cdot \tilde{\mathbf{d}} + (\mathbf{N}^G(\mathbf{x}))^{\mathrm{T}} \cdot \mathbf{g} \tag{21a}$$

$$\frac{\partial \mathbf{u}^K(\mathbf{X})}{\partial \mathbf{X}} = \left(\frac{\partial \mathbf{N}^J(\mathbf{X})}{\partial \mathbf{X}}\right)^{\mathrm{T}} \cdot \tilde{\mathbf{d}} + \left(\frac{\partial \mathbf{N}^G(\mathbf{X})}{\partial \mathbf{X}}\right)^{\mathrm{T}} \cdot \mathbf{g} \tag{21b}$$

where

$$\mathbf{N}^J(\mathbf{X}) = \mathbf{J}^{\mathrm{T}} \cdot \mathbf{N}(\mathbf{X}) \tag{22a}$$

$$\mathbf{N}^G(\mathbf{X}) = \mathbf{G}^{\mathrm{T}} \cdot \mathbf{N}(\mathbf{X}) \tag{22b}$$

Note that the constraint conditions are recovered. That is

$$\mathbf{G}^{\mathrm{T}} \cdot \mathbf{d} = \mathbf{G}^{\mathrm{T}} \cdot (\mathbf{J} \cdot \tilde{\mathbf{d}} + \mathbf{G} \cdot \mathbf{g}) = \mathbf{g} \tag{23}$$

Since the new transformed interpolation functions of Equation (21) satisfy the essential boundary conditions, they can be directly used in the virtual work statement. The alternate displacement vector, $\tilde{\mathbf{d}}$, is now determined with the essential boundary conditions directly specified in $\mathbf{g}$. The orthogonalization in Equations (16)–(18) may require a significant number of computer operations. These computations and the creation of the alternate interpolation functions of Equation (21), however, are done only once in the preprocessing phase. Transient displacement conditions (e.g. specified displacements) are then accommodated by only changing

the values in **g**. Note that this approach may be effective for serial computations. For analyses on a single processor, it might be more beneficial to use the transformation directly on the displacements, instead of the interpolation functions. As shown in Section 4, transformation of the interpolation functions in this manner is convenient for parallel processing.

To use the interpolation form of Equation (21), **g** is simply appended to the end of the displacement vector as dummy degrees of freedom. For insertion into the virtual work statement, the values in **g** are then just treated like other displacements having their own interpolation functions (22b). Note that separate interpolation functions are generally required for each degree of freedom in Equation (21), instead of for each node as in Equation (2). This does not require significantly more computations, however, and the additional storage requirements are not detrimental on most large parallel processing computer systems. In many cases (e.g. zero conditions), the additional degrees may be omitted. In other cases, the conditions may be static or have a simple time variance, so that the second term in Equation (21) may be collapsed into a single dummy node, such that

$$\mathbf{u}^K(\mathbf{x}) = (\mathbf{N}^J(\mathbf{X}))^{\mathrm{T}} \cdot \tilde{\mathbf{d}} + \tilde{\mathbf{g}}(\mathbf{X}) \tag{24}$$

In such cases, the product of the interpolation functions and specified quantities are summed into a single nodal vector. The dummy node interpolation functions are specified as unity, and $\tilde{\mathbf{g}}$ is multiplied by a time factor during the analysis.

In the above method, the EBCs are only specified at discrete points; the EBCs are not generally satisfied in between these points and an integral expression only imposes in an average sense. Nevertheless, the method is effective and the accuracy is increased with model refinement. EBC imposition continues to be a research area with much activity [10–13]. One popular approach is to transform potential essential boundary condition nodes, using the interpolation functions determined at the nodes, to recover the Kronecker delta property at the nodes [13]. This is especially effective for contact problems where the contacting nodes are not known *a priori*. The transformation can be done in a similar manner to that of Equations (14)–(24) and would thus possess a form with similar benefits for parallel processing that is discussed in Section 4.

3.3. Time-integration scheme

A Newmark-β method is used to integrate the governing equations in time. Grouping expressions with common variational degrees of freedom in Equation (10) produces the following discretized expression in matrix form:

$$\mathbf{M}\ddot{\mathbf{d}}^t = \mathbf{p}^t - \mathbf{f}^t \tag{25}$$

where **M** is the mass matrix, **d** is the generalized nodal displacement vector, **p** is the vector of applied and body forces, and **f** is the vector of internal resistance forces. The temporal integration begins by specifying initial conditions and then, for each time increment

(i) Use a predictor phase:

$$\hat{\mathbf{d}}^{t+\Delta t} = \mathbf{d}^t + \Delta t\dot{\mathbf{d}}^t + (\tfrac{1}{2} - \beta)(\Delta t)^2\ddot{\mathbf{d}}^t \tag{26}$$

$$\hat{\dot{\mathbf{d}}}^{t+\Delta t} = \dot{\mathbf{d}}^t + (1-\gamma)\Delta t\ddot{\mathbf{d}}^t \tag{27}$$

(ii) Update accelerations:

$$\ddot{\mathbf{d}}^{t+\Delta t} = \mathbf{M}^{-1}(\mathbf{p}^{t+\Delta t} - \mathbf{f}^{t+\Delta t}) \tag{28}$$

(iii) Use a corrector phase:

$$\mathbf{d}^{t+\Delta t} = \hat{\mathbf{d}}^{t+\Delta t} + \beta\,(\Delta t)^2\,\ddot{\mathbf{d}}^{t+\Delta t} \tag{29}$$

$$\dot{\mathbf{d}}^{t+\Delta t} = \hat{\dot{\mathbf{d}}}^{t+\Delta t} + \gamma \Delta t\,\ddot{\mathbf{d}}^{t+\Delta t} \tag{30}$$

(iv) Continue with the next time increment.

The particle/node mass is lumped, so that the mass matrix, **M**, is diagonal. The computations associated with Equations (26)–(30) are primarily vector operations. Therefore, CPU usage is dominated by the determination of **f** in Equation (28), which frequently involves intensive computations of $\boldsymbol{\sigma}$ for inelastic constitutive relationships.

4. PARALLEL CODE DEVELOPMENT

For effective parallel computing, it is critical to balance the computational load among processors while minimizing interprocessor communication. Therefore, separate pre-analysis software was created to partition any general unstructured RKPM model. Similar to explicit finite element codes [14–16], the partitioning is made with regard to the determination of **f** in Equation (28), since it involves more computational effort than the lumped mass equation solving. Therefore, integration points are distributed to processors and nodes are shared by integration points on different processors. In contrast to finite elements, the amount of computations will vary among integration points, since each may contribute to a different Number of Particles/nodes (NP). Depending on the levels of local refinement, NP may vary drastically within the model. Unlike most Smooth Particle Hydrodynamics (SPH) formulations that do not distinguish between particles and integration points, the present RKPM models usually have many more integration points than particles/nodes. Whereas parallel SPH implementations [15] typically partition the particles, the RKPM partitioning is thus done on the integration points. The objective of the partitioning software is to provide partitions of nearly equal computational effort while also generally minimizing the size of the partition interfaces. An outline of the overall analysis procedures is provided in Table I.

4.1. Partitioning schemes

A variety of different partitioning procedures exist that are based on graph theory (e.g. [17–19]) or geometric techniques (e.g. [19–21]). Graph based procedures are quite popular for finite element analysis whereby they are generally accurate and efficient. Furthermore, publicly available partitioning codes like Metis [17], Jostle [18] and Chaco [19] have matured to the point where they can be easily and reliably used as 'black boxes' for most finite element applications. Graphs for the present RKPM implementation are the integration point to integration point connection lists. In graph theory terminology, each integration point is called a vertex. The list of other integration points to which the vertex is connected is then defined as a list of edges to other vertices.

Each RKPM integration point has a list of support particles/nodes within its region of influence that it must provide a contribution to the particle/nodal forces of Equation (28). This list is used to create the graph edges by identifying integration points with common support nodes. For the graph, vertex weighting permits the specification of variable amounts of computational

Table I. Outline of procedures for parallel computation of RKPM explicit dynamic analysis.

I. Serial pre-analysis phase
  A. Input data
  B. Set up Gauss integration points
  C. Perform global search for support nodes
  D. Calculate lumped mass matrix
  E. Partition model
    1. Create reduced graph of integration points
    2. Use Metis to produce integration point processor list
    3. Create local model input for each processor
      a. List of partition integration points
      b. List of partition nodes
      c. Processor communication lists of duplicate nodes
      d. Materials, loads, etc., applicable to partition

II. Parallel analysis phase on each processor
  A. Read input data for the appropriate partition
  B. Setup data structure for local analysis of partition
  C. Read lumped mass matrix for nodes within partition
  D. Calculate shape functions and their derivatives
  E. Calculate initial accelerations, $\ddot{\mathbf{q}}^0$
  F. Begin time-incrementation loop
    1. Post-non-blocking receives for force vectors of duplicate nodes;
    2. Determine predictor quantities in Equations (26), (27);
    3. Compute applied forces and place in $\mathbf{p}^{t+\Delta t}$;
    4. Calculate $\mathbf{f}^{t+\Delta t}$ and store as $\mathbf{p}^{t+\Delta t} = \mathbf{p}^{t+\Delta t} - \mathbf{f}^{t+\Delta t}$;
    5. Gather duplicate node force vectors from $\mathbf{p}^{t+\Delta t}$ and post-non-blocking sends;
    6. Post non-blocking waits for completion of communications in steps 1 and 5;
    7. Add received force vectors contributions from other processors to $\mathbf{p}^{t+\Delta t}$;
    8. Determine $\ddot{\mathbf{q}}^{t+\Delta t}$, $\dot{\mathbf{q}}^{t+\Delta t}$, and $\mathbf{q}^{t+\Delta t}$ via Equations (28)–(30);
    9. Output, if timely
    10. For the total time, repeat steps 1–9 with new configurations.

III. Serial post-analysis phase
  A. Read common output files from each processor
  B. Create a single output file for post-processing

effort associated with each vertex. Edge weighting specifies the amount of communication associated with each edge. For the present RKPM implementation, vertex weighting is applied according to the number of nodes to which an integration point contributes. All graph edge weights are defined to be the same. The goal of a graph-based partitioner is to distribute the vertices so that each partition has equal amounts of vertex weight, while minimizing the amount of edge weight connecting partitions on different processors. The quality of a RKPM partitioning is thus defined by the balance of vertex weights and by the number of nodes shared by integration points on different processors.

RKPM integration points typically contribute to many more nodes than those of similar finite element models do. Thus, RKPM graphs can be very large with many edges. For example, the graph of the simple problem in Figure 2 has an average of 175 edges per vertex and the graph file requires almost 120 Mbytes. Using Metis [17], a typical partition could be obtained for this

model is about 2.5 CPU minutes on an SGI R10000 processor. Although the Metis partitions were quite good and the amounts of CPU time were reasonable for the example of Figure 2, this approach may be unfeasible for large three-dimensional models.

To reduce the number of edges with RKPM, an approach using simple geometric checks was developed for creation of the Metis graph. Only the nearest neighbours (edge members) of each integration point (vertex) were included in the graph. This should not significantly affect the quality of the partitions, since the nearest neighbours have the greatest level of connection. First, a radius of influence from the vertex points is input to initially remove, from consideration as edges, points that lie outside this radius. Further reduction may then be applied by retaining only a percentage of the closest neighbours. A minimum number of edges, however, are used to ensure adequate connections within the graph. Retaining only 10 per cent of the original number of edges with a minimum of four, the graph for the model of Figure 2 was reduced to about 5 Mbytes and excellent partitions were determined with Metis in about 2 CPU seconds on the R10000.

To avoid the creation of large dense graphs, geometric-based partitioners seem to be natural choices for RKPM. These approaches do not need a graph, as they only require the RKPM integration point co-ordinates. Partitions are created with these methods by grouping points in the same spatial proximity. The authors' experiences with existing Recursive Coordinate Bisection (RCB), Hilbert Space Filling Curve (HSFC), and Unbalanced Recursive Bisection (URB) codes, however, have not always been satisfactory. In some instances, large imbalances (up to 20 per cent) were noticed. For all cases, Metis provided partitions that were perfectly or almost perfectly balanced. This may be more of a problem with the software implementation, instead of the approach itself. Space filling curves created solely by geometry, for instance, may inherently exhibit balance difficulties, if a large weight is near a partition boundary on the curve. The weight significantly contributes too much if the point is added to the partition, but the partition will be considerably under-weighted if it is not kept within the partition. In either case, a large imbalance will occur when division of the curve is into equal weighted lengths without redefining the curve under these circumstances. Early Metis releases also experienced similar difficulties for highly variable vertex weighting, but significant effort has been put into Metis to overcome these problems. The authors did not attempt to alleviate these problems with the other existing codes. Furthermore, difficulties arise for applications like the one in Figures 3–5. Although integration points on either side of the notch are spatially close to each other, they do not have common support nodes. The geometric partitioners typically grouped points on both sides of the notch within the same partition, which occasionally increased the number of shared interpolation nodes significantly. For these reasons, the forgoing graph-based method is currently used with the present RKPM implementation.

4.2. Parallel data-communication structure

The partitioning output provides a list of processor numbers for the integration points, so that each is uniquely defined on a single processor. The diagonal nature of the mass matrix, **M**, permits the particle/nodal equation of each degree of freedom to be solved independent of other degrees of freedom. To retain data locality, particles/nodes are therefore redundantly defined on all processors possessing integration points contributing to these particles/nodes. An example of the partitioning of integration points and nodes is shown in Figures 4 and 5. Note that the Metis partitioning occurs naturally without shared nodes along the notch. As a result of the nonlocal nature of the interpolation in Equation (2), duplicate nodal definitions generally do not only lie

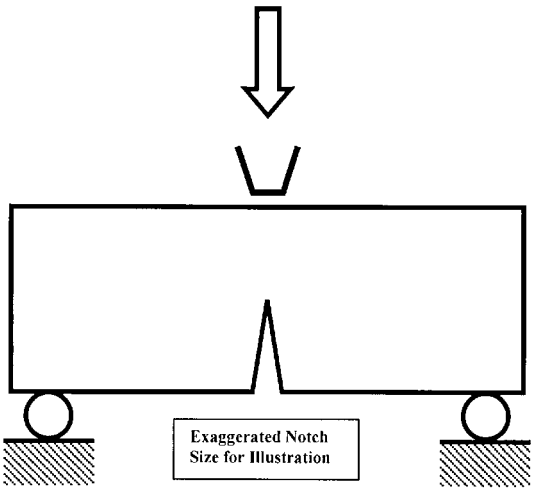

Figure 3. Notched three-point bending problem.

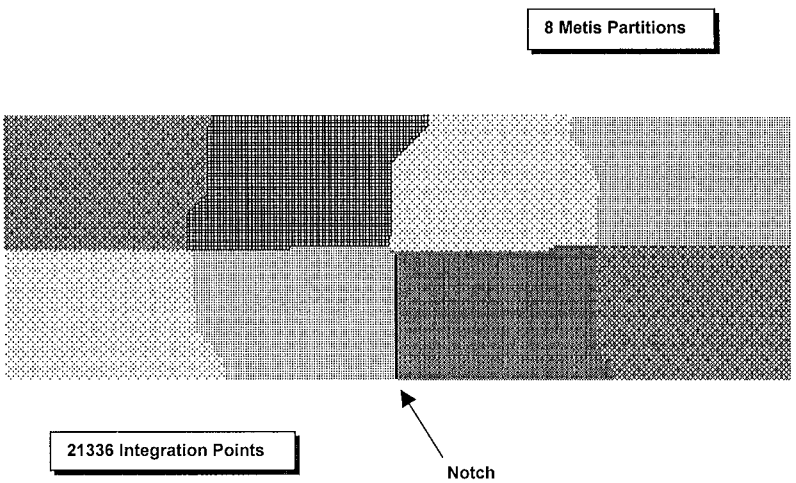

Figure 4. Partitioned RKPM model (integration points) of notched three-point bending problem.

directly on partition boundaries. As seen in Figure 5, a band of shared nodes typically lies at partition interfaces. Therefore, the amount of communication will generally be larger for RKPM than for similar finite element analyses. The present RKPM implementation, however, uses more integration points and support nodes than FEA, so that the computation to communication ratio is still high.

The basic parallel approach is for each processor to perform an analysis on its own partition. These are performed as if they were separate analyses, except that contributions to the force

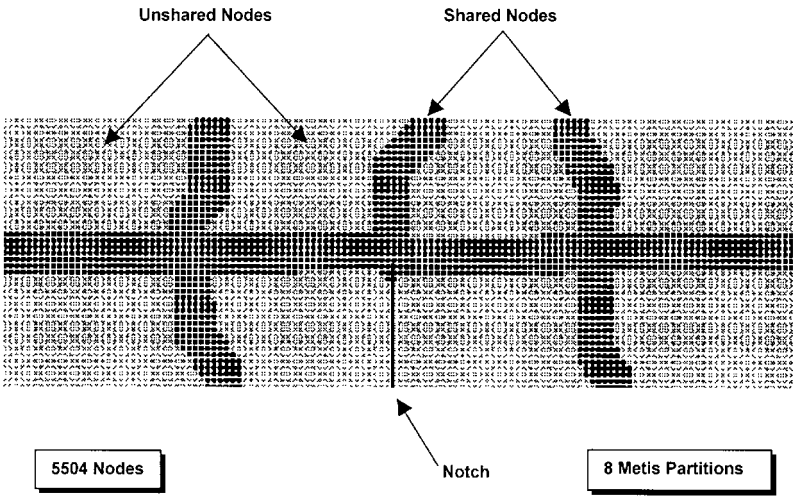

Figure 5. Shared and unshared nodes for the partitioned RKPM model in Figure 4.

vectors are sent to and from other processors for the duplicated nodes. All loads, boundary conditions, material properties, etc., need to be defined only on the processors for which they apply. The Lagrangian formulation of Section 3.1 is well suited for parallel processing. With the support nodes and interpolation functions fixed to the reference configuration, the interprocessor communication structure does not change during a transient analysis. An advantage to this computational structure is that an original serial code can be made to run in parallel with only slight modifications. Although a considerable amount of code is written for the partitioned analysis procedures, it is mostly added to the serial program, which essentially remains intact.

All communication is made by explicit Message Passing Interface (MPI) statements. Non-blocking communication (MPI ISEND/IRECV) is used to avoid possible deadlocks and overhead associated with buffering. For each time increment, Δt, the parallel scheme first consists of creating global force vectors for the partitions on each processor, $\mathbf{p}^{t+\Delta t}$ and $\mathbf{f}^{t+\Delta t}$. Next, the forces belonging to redundant particle/nodes are gathered into vectors and sent to the processors possessing duplicate definitions. The partial force vectors are then received from the other processors and added to the global partition force vector on the current processor. At this point, other boundary conditions are accommodated in the force vector, and the new accelerations, velocities, and displacements are determined by the relations in Equations (26)–(30). Using the new configuration, the process is then started all over again for a new time increment. When the analysis is finally completed, separate software is run serially to gather the individual processor output and create single output files for post-processing.

4.3. Comments on essential boundary condition enforcement

The approach described in Section 3.2 maintains the original RKPM interpolation, but also requires some additional computations. The orthogonalization in Equations (16)–(18), however, is only done once in the pre-analysis phase. All necessary quantities can then be entirely defined

on each processor, so that the imposition during the transient analysis can be performed locally without inter-processor communication. That is, the support nodes are redefined to accommodate the interpolation in Equation (21) and then, the partitioned data sets are created in the usual manner described in Sections 4.1 and 4.2.

The effectiveness of approach described in Section 3.2 is highly problem dependent. For many applications, the essential boundary conditions are minimal and the approach is highly efficient. For problems with a large number of essential boundary conditions, however, the number of support nodes can significantly increase for integration points affected by the modifications in Equation (21). The vertex weighing by number of support nodes, as described in Section 4.1, will avoid associated load imbalances, but the number of shared nodes can drastically increase. Therefore, communications may greatly increase thereby reducing the number of effective processors, in some cases. Further effort regarding the accommodation of essential boundary conditions in parallel is still needed, and thus several other techniques are discussed here.

Projecting integration points across symmetry lines, like those in Figure 2, can accommodate symmetry conditions [22]. Contributions to particles/nodes are complete for the symmetric portion of the model, which enforces the symmetry without equations solving. Additional vertex weighing can be applied to the projected points, in order to account for the additional computational load of these points. Although this approach reduces the number of simultaneous equations needed to enforce the essential boundary conditions, additional computations result from the additional integration. Nevertheless, the integration calculations are entirely local and are thus better suited for parallel processing. This procedure only addresses some of the conditions, however, since non-zero essential boundary and initial conditions must be accommodated in a different manner.

The method by Chen [10, 11] introduces singularities into the kernel functions of essential boundary condition particles, in order to recover particle values. The interpolation still maintains the reproducing characteristic of RKPM and the singularity is only at the particles, not at integration points. The evaluation of the governing equations is thus not made at the singularities. The modified kernel functions permit kinematic constraints to be imposed directly at the particles, which can be done locally on a processor and without any significant computational cost that would affect load balance. Whereas reasonable results were obtained for the examples by Chen [10, 11], the modified kernel function does affect the solution (the interpolant is modified). The use of the singular kernels may prove to be a viable approach, but the authors believe that more evaluation of the method is still warranted.

5. NUMERICAL EXAMPLES

5.1. Three-point bending of a notched beam

The following analyses are applied to the three-point flexural analysis of a notched beam, as shown in Figure 3. For the temporal integration, the factors in Equations (26)–(30) are $\gamma = 1.0$ and $\beta = 0.0$. A viscoplastic constitutive model with damage was used and details of the analyses are described in Reference [23].

The first RKPM model uses 5504 nodes and 21 336 integration points. The original graph of the model averaged approximately 500 edges per vertex. Using the reduction scheme described in Section 4.1, the graph was reduced to about 10 edges per vertex. The partitioning is shown in

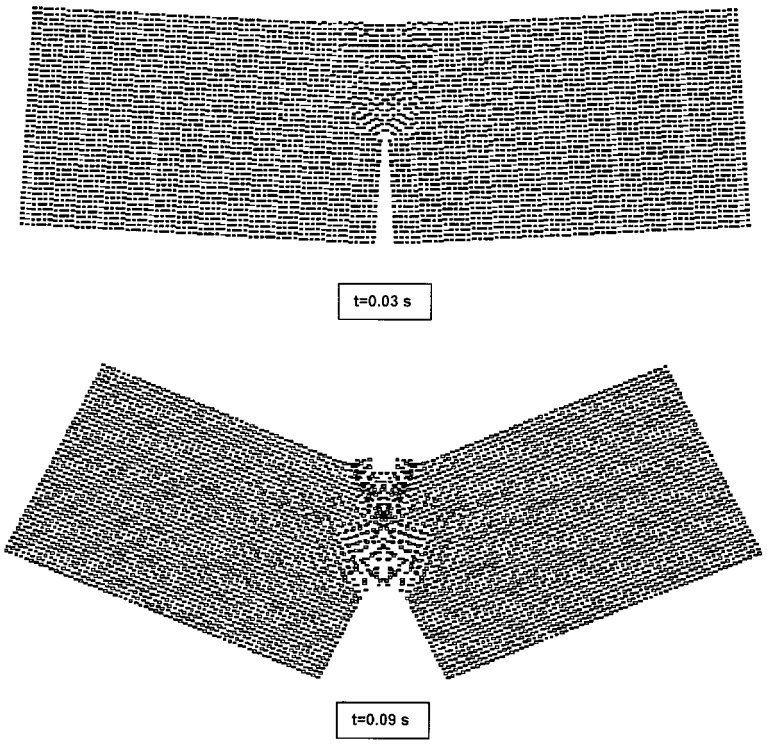

Figure 6. Predicted RKPM deformations of notched three-point bending problem (5504 nodes).

Figures 4 and 5, and deformed plots of the RKPM model are shown in Figure 6. The parallel performance is given in Figure 7 for analyses on the Cray T3E-1200 at the Army High Performance Computing Research Center. The speedup is significant for this moderately sized problem. A speedup of almost fifteen was attained on 32 processors which was about the peak value that could be attained. An analysis that took about fifteen hours on a single processor was thus performed in less than one hour by parallel processing. In all cases, no significant reduction in performance occurred by using the reduced graph partitioning instead of the full graph partitioning.

The second RKPM model is larger with 21 590 nodes and 85 008 integration points. Plots of the effective plastic strain and deformed shape of the RKPM model are shown in Figure 8. The parallel performance is given in Figure 9 for analyses on the Cray T3E-1200. These analyses require about six CPU hours on 128 processors. The high level of refinement permits the sharp resolution prediction of the shear bands shown by the light areas of high effective plastic strain in Figure 8. For economical reasons, the scalability study depicted in Figure 9 was only performed with 100 000 time increments and with no less than four processors. Therefore, the scalability is made with reference to four processors (speedup from the four processor analysis). The speedup is significant for this large model. The analysis on 128 processors was about seventeen times faster than the one on four processors. Using these factors as a guideline, the analysis for Figure 8 that took about six hours on 128 processor would require more than four days on four processors.

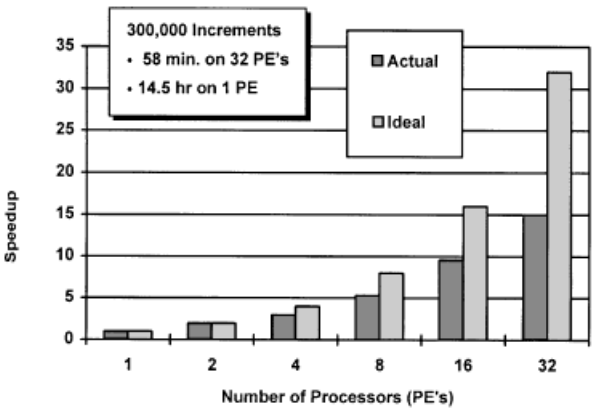

Figure 7. Parallel performance on a Cray T3E-1200 of RKPM for the small notched three-point bending problem (5504 nodes).

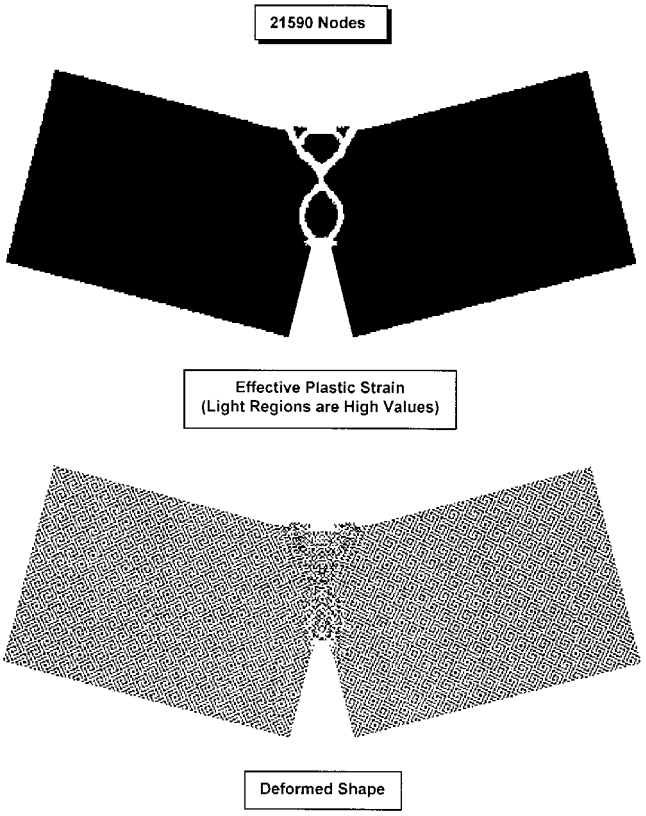

Figure 8. RKPM predictions for notched three-point bending problem at $t = 0.06$ s (21 590 nodes).

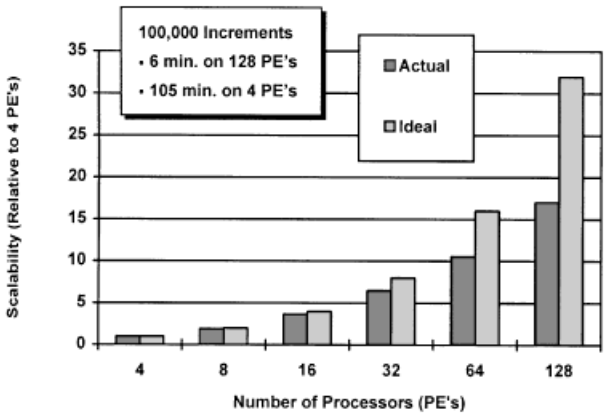

Figure 9. Parallel performance on a Cray T3E-1200 of RKPM for the notched three-point bending problem (21 590 nodes).

Furthermore, this analysis was also performed on a single dedicated SGI R10000 processor, which required more than seventeen days of CPU time.

5.2. Three-dimensional shear band simulation in a tensile specimen

This analysis is the three-dimensional development of shear bands in the tensile specimen shown in Figure 10. The undeformed bar has cross-section dimensions of 2 mm × 2 mm and a length of 4 mm. An elasto-viscoplastic constitutive model was used and, for the temporal integration, the factors in Equations (26)–(30) are also $\gamma = 1.0$ and $\beta = 0.0$. The RKPM model uses 18 081 nodes and 128 000 integration points, shown in Figure 10. The original graph of the model averaged approximately 700 edges per vertex. Using the reduction scheme described in Section 4.1, the graph was reduced to about 50 edges per vertex.

A plot of the deformed shape, depicting the shear bands, for the RKPM model is also shown in Figure 10. For an 8×10^{-5} s duration event, the analysis required about 400 CPU seconds on 256 processors of the Cray T3E-1200 at Waterways Experiment Station. The parallel performance is given in Figure 11 for the 10 000 time increment analysis. For economical reasons, this scalability study was again performed with no less than four processors. Therefore, the scalability is made with reference to four processors. The speedup is significant for this model. The analysis on 256 processors was about 47 times faster than the one on four processors.

6. CONCLUDING REMARKS

A parallel computational implementation of modern meshless methods was presented for explicit dynamic analysis. Application of the Reproducing Kernel Particle Method demonstrated the procedures. A coarse grain parallel implementation was used with model partitioning for

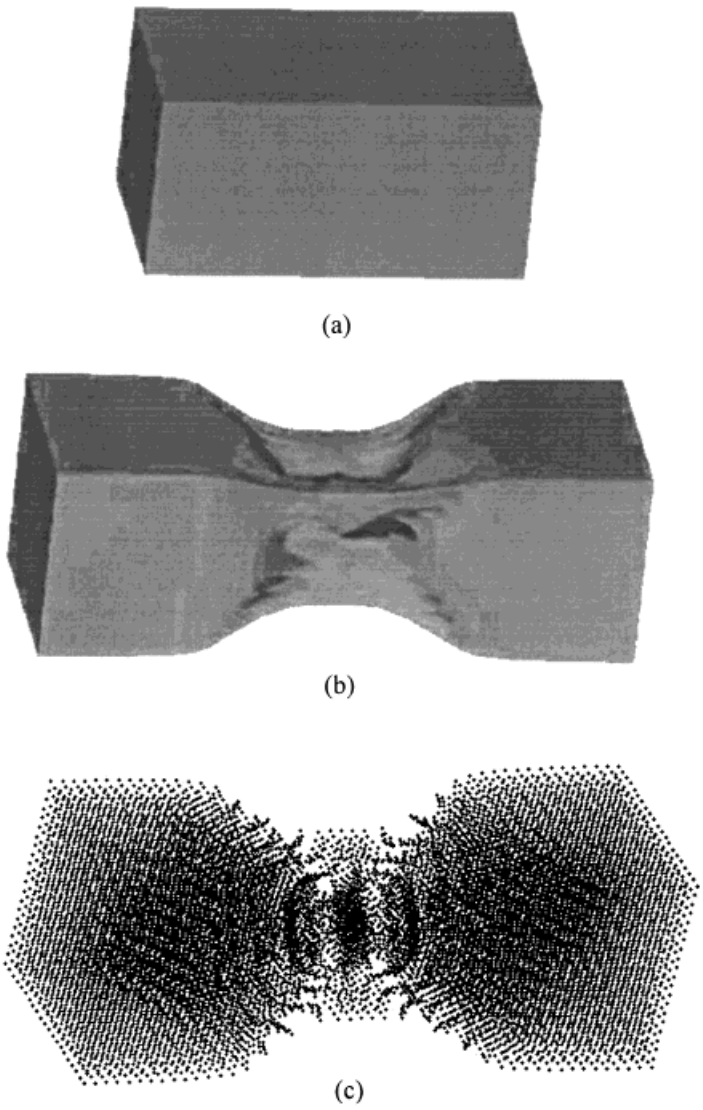

Figure 10. RKPM model for the three-dimensional tensile shear band development: (a) undeformed shape; (b) deformed shape at $t = 8 \times 10^{-5}$ s; (c) deformed particles at $t = 8 \times 10^{-5}$ s (18 081 nodes).

a Lagrangian formulation. With this approach, integration points are uniquely defined on separate processors and shared particle definitions are duplicated, so that all support particles for each point are defined locally on the corresponding processor. Several partitioning schemes were evaluated and a reduced graph-based procedure was shown to be effective. Explicit MPI message passing statements are used for all communications among partitions on different processors. Procedures to accommodate essential boundary conditions in parallel were presented. It was

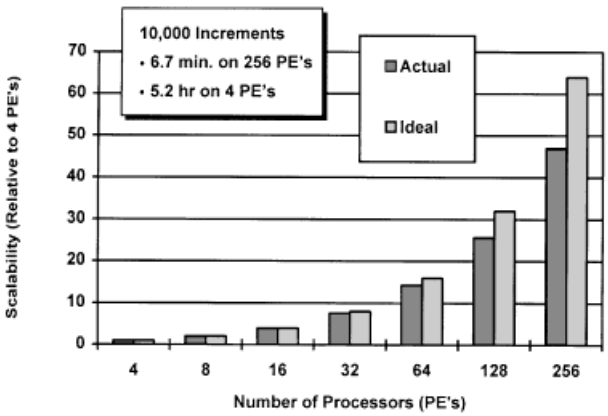

Figure 11. Parallel performance on a Cray T3E-1200 of RKPM for the three-dimensional tensile shear band development problem (18 081 nodes).

pointed out that the enforcement of essential boundary conditions in parallel merits more investigation. Nevertheless, the procedure was demonstrated to significantly reduce computation time for several highly deformable inelastic analyses.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the technical suggestions of Dr. George Karypis at the Army High Performance Computing Research Center, University of Minnesota. The work was supported in part by grants of HPC time from the DoD HPC Center at U.S. Army Engineer Waterways Experiment Station and from the Army High Performance Computing Research Center. This work is sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory co-operative agreement number DAAH04-95-2-003/contract number DAA-95-C-0008, the content of which does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred. R. A. Uras' work was supported by the U.S. Department of Energy, Technology support Program, under contract W-31-109-Eng-38.

REFERENCES

1. Lucy LB. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal* 1977; **82**(12):1013–1024.
2. Belytschko T, Lu YY, Gu L. Element free Galerkin methods. *International Journal for Numerical Methods in Engineering* 1994; **37**:229–195.
3. Duarte CA, Oden JT. Hp clouds—a meshless method to solve boundary-value problems. *Technical Report 95-05*, Texas Institute for Computational and Applied Mathematics, University of Texas, Austin, 1995.
4. Liu WK, Jun S, Li S, Adee J, Belytschko T. Reproducing kernel particle methods for structural dynamics. *International Journal for Numerical Methods in Engineering* 1995; **38:**1655–1679.
5. Liu WK, Chen Y, Uras RA, Chang CT. Generalized multiple scale reproducing kernel particle methods. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:91–158.
6. Chen JS, Pan C, Wu CT, Liu WK. Reproducing kernel particle methods for large deformation analysis of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering* 1996; **139**:195–228
7. Babuska I, Melenk JM. The partition of unity finite element method. *University of Maryland Technical Note BN-1185*, 1995.

8. Jun S, Liu WK, Belytschko T. Explicit reproducing kernel particle methods for large deformation problems. *International Journal for Numerical Methods in Engineering* 1998; **41**:137–166.
9. Malvern LE. *Introduction to the Mechanics of a Continuous Media*. Prentice-Hall: Englewood Cliffs, NJ, 1969.
10. Chen JS. Improved mesh-free method for incompressible boundary value problems, *Presented at the Fourth World Congress on Computational Mechanics*, Buenos Aires, Argentina, 29 June–2 July 1998.
11. Chen JS, Wang H-P. New boundary condition treatments in meshfree computation of contact problems. *Computer Methods in Applied Mechanics and Engineering*, accepted for publication.
12. Günther FC, Liu WK. Implementation of boundary conditions for meshless methods. *Computer Methods in Applied Mechanics and Engineering* 1998; **163**(1-4):205–230.
13. Chen JS, Pan C, Roque CMOL, Wang HP. A Lagrangian reproducing kernel particle method for metal forming analysis. *Computational Mechanics* 1998; **22**:289–307.
14. Hoover CG, DeGroot AJ, Maltby JD, Procassini RJ. ParaDyn—DYNA3D for massively parallel computers. Engineering, Research, Development and Technology FY94. *UCRL 53868-94*, Lawrence Livermore National Laboratory, Livermore, CA, 1995.
15. Plimpton S, Attaway S, Hendrickson B, Swegle J, Vaughan C, Gardner D. Transient dynamics simulations: parallel algorithms for contact detection and smoothed particle hydrodynamics. *Proceedings of SuperComputing 96*, Pittsburgh, PA, 1996.
16. Danielson KT, Namburu RR. Nonlinear dynamic finite element analysis on parallel computers using FORTRAN 90 and MPI. *Advances in Engineering Software* 1998;**29**(3-6):179–186.
17. Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *Technical Report TR 95-035*, Department of Computer Science, University of Minnesota, 1995.
18. Walshaw CH, Cross M, Everett MG. A localized algorithm of optimizing unstructured mesh partitions. *International Journal of Supercomputer Applications* 1995; **9**(4):280–295.
19. Hendrickson B, Leland R. The Chaco user's guide: version 2.0. *Technical Report SAND94-2692*. Sandia National Labs, Albuquerque, NM, 1995.
20. Sagan H. *Space Filling Curves*. Springer: Berlin, 1994.
21. Jones MT, Plassmann PE. Computational results for parallel unstructured mesh computations. *Computing Systems in Engineering* 1994; **5**(4-6):297–309.
22. Libersky LD, Petschek AG. Smooth particle hydrodynamics with strength of materials. In *Advances in the Free-Lagrange Method*, Lecture Notes in Physics, Trease HE *et al.* (eds). Springer: Berlin, 1990.
23. Hao S, Liu WK, Chang CT. Computer implementation of damage models by finite element and meshfree methods. *Computer Methods in Applied Mechanics and Engineering*, accepted for publication.

