CSMA 2019
14ème Colloque National en Calcul des Structures
13-17 Mai 2019, Presqu'île de Giens (Var)

The Meshfree Explicit Galerkin Analysis (MEGA) Code

M. Hillman[1], K.C. Lin[1], A. Madra[1]

[1] *Department of Civil and Environmental Engineering, The Pennsylvania State University,*
{mhillman,kul218,annamadra}@psu.edu

Résumé — For extremely large deformation problems in solid mechanics, the Lagrangian finite element approach is often ineffective. Galerkin meshfree methods developed extensively over the past twenty years and are sufficiently mature to handle simulations involving large deformations with relative ease. Many state-of-the-art advances in these methods have been implemented into a research code called *Meshfree Explicit Galerkin Analysis* (MEGA) at The Pennsylvania State University. The code has been validated and verified against both linear and nonlinear mechanics benchmark solutions.
Mots clés — meshfree methods, finite strain, explicit dynamics.

1 Introduction

Galerkin meshfree methods offer a path forward to compute solutions to extremely large deformation problems where the traditional finite element approach is generally ineffective [1]. A meshfree discretization is simply a point-cloud of nodes and surface information and constructs shape functions directly in Cartesian coordinates. Thus, a meshfree approach and does not rely on a mesh to construct approximation functions; this difference between the traditional finite element method is contrasted in Figure 1 [1]. Correspondingly, these methods offer significant advantages in extremely large deformation problems where Lagrangian finite elements become distorted or entangled. This is made possible by the fact that node connectivity and shape functions can be continually reconstructed, rather than being dictated by a Lagrangian element topology.

The Meshfree Explicit Galerkin Analysis (MEGA) code is an explicit, large-strain nonlinear dynamic code based on the meshfree Reproducing Kernel Particle Method (RKPM) [2, 3]. The reproducing kernel approximation is employed for test and trial functions resulting in a Bubnov-Galerkin discretization in space under the Updated Lagrangian formulation [4]. In order to handle extreme deformations where the mapping between undeformed configuration and current configuration is no longer valid, a semi-Lagrangian RKPM discretization [5] is employed which constructs the meshfree approximations in the current configuration.

Nodal integration is employed for (spatial) domain integration, i.e., meshfree particles (or nodes) are used as the integration points themselves. Thus, the meshfree particles serve as Lagrangian material points, allowing natural treatment of path-dependent material models. A strain-smoothing method called Stabilized Non-conforming Nodal Integration (SNNI) [5] is adopted in MEGA in order to remedy the

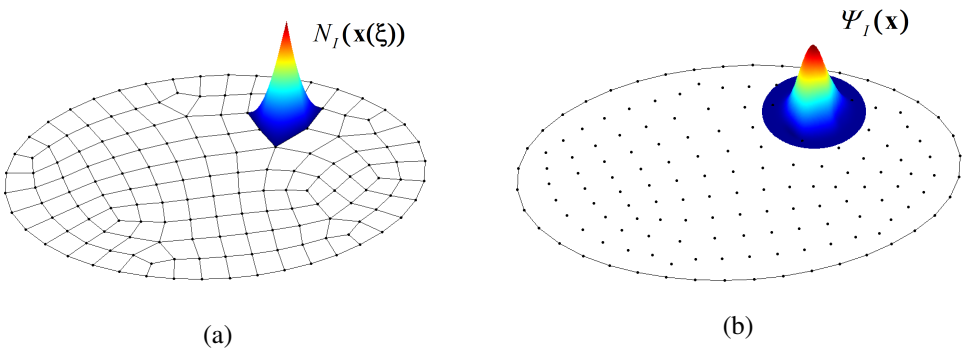


FIGURE 1 – (a) A Finite Element shape function in the global domain and (b) Meshfree shape function.

instability inherent in nodal integration and to provide improved solution accuracy over direct nodal integration. Naturally Stabilized Nodal Integration [6] is further introduced under the strain-smoothing framework, which provides additional coercivity in nodal integration.

With the meshfree spatial discretization in hand, time-space calculations are performed using the Newmark-β time integration scheme [7] with the central difference method. To make the algorithm fully explicit, a lumped mass matrix is utilized, along with mass-proportional damping.

The remainder of this paper is organized as follows. First, the code structure is discussed in Sec. 2 along with user input and code execution. Governing equations and the meshfree discretization of the equations are given in Sec. 3. Numerical quadrature via enhanced nodal integration is discussed in Sec. 4, and time integration of the semi-discrete form is discussed in Sec. 6. Constitutive modeling is treated in Sec. 5, and numerical examples are given in Sec. 7. Finally, a summary of current capabilities and limitations, and future directions are given in Sec. 8.

2 Computer Implementation

2.1 MEGA Code Structure

MEGA is a parallel Fortran90 code which employs Open Multi-Processing (OpenMP) for a shared memory implementation, suitable for running in a workstation environment. The code structure follows the algorithms outlined in Sections 3-6, and is summarized in Algorithm 1.

Data: Model and input parameters : Control.dat and *.k (Section 2.2)
Result: VTK file with nodal data requested in the control file (Section 2.3)
Initialize OpenMP;
Generate nodal volumes, supports, and smoothing cells;
Assign boundary conditions, material properties and other attributes to nodes;
Log model, control parameters, and nodal properties;
while *time* $<$ *total simulation time* **do**
  Form predictors (Section 6);
  Do node search;
  for *all selected nodes* **do**
    Pick up nodal data from global arrays;
    Construct shape functions (Section 3);
    Calculate strain (Section 4);
    Calculate stress and assemble the contribution of this node to the internal force (Section 5);
  end
  Assemble the contribution of this thread to the global internal force;
  Generate physical displacements from generalized displacements;
  Update nodal positions;
  Calculate acceleration from equation of motion;
  Calculate correctors and assign boundary conditions (Section 6);
  if *output timer has passed the output frequency* **then**
    Output data;
    Reset the timer;
  end
  Log the estimates of total simulation time remaining;
end

Algorithm 1: Structure of the MEGA program

2.2 MEGA Input and Execution

Two files are required for running MEGA. The first file called *the model file* provides the geometric definition of the problem along with the identification of sets of nodes. This file has an LS-DYNA input

```
*MODEL                              #SET NO (in CUBIT or TRELIS)    BAR
# Name                              2                               #TYPE
taylorbar.k                         #NAME OF THIS SET               2
                                    WALL                            #props
*TIME PARAMETERS                                                    0.3 78.2e9 2700 0.29e9 125
# TIME_END                          *BOUNDARY CONDITION             0.1 0 0 0 0
1.0E-04                             #SET NO. OR NODE SET NAME       0 0 0 0 0
#IAUTO_TS (AUTO TIME STEP = 1)      WALL                            0 0 0 0 0
1                                   #FIXITY                         0 0 0 0 0
# FACTOR                            0 0 1                           0 0 0 0 0
0.9
                                    *INITIAL CONDITION              *OUTPUT PARAMETERS
*SET NAME                           #SET NO. OR NODE SET NAME       #TIME_OUTPUT
# SET NO (in CUBIT or TRELIS)       BAR                             5.0E-07
1                                   #VEL
#NAME OF THIS SET                   0 0 -373                        *OUTPUT VARIABLES
BAR                                                                 basic
                                    *MATERIALS
*SET NAME                           #SET NO. OR NODE SET NAME
```

FIGURE 2 – Example of a Control File for MEGA.

file format and is stored as '*.k' (* here indicates a wildcard). Information on the exact format of this file is readily available online, and can also be generated with a variety of preprocessors.

The second one is the *Control File* named '*Control.dat*', which is a text file containing all the other information needed to run the simulation, such as specification of boundary conditions, material properties, total simulation time, and so on. These aspects are specified in terms of node sets unless they are global parameters such as output frequency and simulation time. The node sets are assigned when building the model in the preprocessor and can then be given names in the control file.

The MEGA Control File uses the *card* system. In MEGA, each card begins with an asterisk '*' and is followed directly by the name of the card. Followed by that, are either comment lines or lines giving the specified values. Other than the format of the cards themselves, the Control File is free, i.e., the order of the cards is arbitrary, and apart from mandatory cards (such as the card that specifies total simulation time), certain cards are optional (e.g., cards for advanced users). An example Control File for the Taylor bar problem in Section 7.1 is given in Figure 2.

2.3 Output

The user specifies which variables to output in the Control File under the card '*OUTPUT VARIABLES' with the output frequency specified in '*OUTPUT PARAMETERS'. Many variables are available such as stress, strain, plastic strain, and so on. Output files are in a VTK format [8] using the *UNSTRUCTURED_GRID* data type and are generated for each output step. The content of these files can be rendered with the open-source software ParaView (https ://www.paraview.org/).

3 Spatial Discretization

3.1 Updated Lagrangian Scheme

Let $\Omega_{\underline{X}}$ denote the initial configuration of the body with material coordinates $\underline{X}$ and a boundary $\partial\Omega_{\underline{X}}$, and let $\Omega_{\underline{x}}$ be the current configuration of the body with current coordinates $\underline{x}$ and a boundary $\partial\Omega_{\underline{x}}$ at a time t. The updated Lagrangian equation of motion can be derived using the principle of virtual power in the current configuration [4] :

$$\int_{\Omega_{\underline{x}}} \underline{\nabla}_{\underline{x}} \delta\underline{v} : \underline{\underline{\sigma}} d\underline{x} - \int_{\Omega_{\underline{x}}} \delta\underline{v} \cdot \underline{b}\rho d\underline{x} - \int_{\partial\Omega^h_{\underline{x}}} \delta\underline{v} \cdot \underline{h} d\underline{x} + \int_{\Omega_{\underline{x}}} \delta\underline{v} \cdot \dot{\underline{v}}\rho d\underline{x} = 0, \tag{1}$$

where δ is the variational operator, $\underline{\nabla}_{\underline{x}}$ the left gradient with respect to the current coordinates $\underline{x}$, $\underline{v}$ the material velocity, σ the Cauchy stress, $\underline{b}$ the prescribed body force in the current configuration, ρ the density in the current configuration, $\dot{()}$ denotes differentiation with respect to time, and $\underline{h}$ is the prescribed traction on boundary of the body in the current configuration $\partial\Omega^h_{\underline{x}}$.

3.2 Semi-Lagrangian Reproducing Kernel Approximation

Let the domain $\bar{\Omega}_{\underline{x}} = \Omega_{\underline{x}} \cup \partial\Omega_{\underline{x}}$ be discretized by a set of N_p nodes $\mathcal{N} = \{\underline{x}_1, \cdots, \underline{x}_{N_P} | \underline{x}_I \in \bar{\Omega}_{\underline{x}}\}$ with corresponding node numbers $\mathcal{Z} = \{I | \underline{x}_I \in \mathcal{N}\}$. In the semi-Lagrangian scheme [5], the nodal locations in the current configuration follow the motion of the body, i.e., $\underline{x}_I = \underline{x}(\underline{X}_I, t)$ where $\underline{X}_I$ are nodal locations in the undeformed configuration, but shape functions are constructed with respect to distance measurements in the current configuration.

In conjunction with the Updated Lagrangian Scheme, this avoids any mapping between the current and undeformed configuration, which is invalid in the presence of extremely large deformations. A typical example is when free surface formulation or closure occurs, and the mapping is no longer one-to-one.

The n^{th} order semi-Lagrangian reproducing kernel (RK) approximation of the displacement field $d(\underline{x},t)$ is constructed as [5] :

$$\underline{d}^h(\underline{x},t) = \sum_{I\in\mathcal{Z}} \Psi_I(\underline{x})\underline{d}_I(t) \tag{2}$$

where $\{\Psi_I(\underline{x})\}_{I\in\mathcal{Z}}$ is the set of RK shape functions, and $\{\underline{d}_I(t)\}_{I\in\mathcal{Z}}$ are the associated coefficients. It is important to note that in general, the RK shape function lacks the Kronecker delta property ($\Psi_I(\underline{x}_J) \neq \delta_{IJ}$), and thus the coefficients are not the actual values of displacements at the nodes, and are termed *generalized displacements*. This results in difficulty in imposing essential boundary conditions. In MEGA, the boundary singular kernel technique [9] is adopted where nodal coefficients on the boundary take on the value of their associated field variables, and conditions are enforced directly at the nodes.

The shape functions are constructed by the product of a kernel function $\Phi_a(\underline{x}-\underline{x}_I)$ and the correction function $C(\underline{x};\underline{x}-\underline{x}_I)$:

$$\Psi_I(\underline{x}) = \Phi_a(\underline{x}-\underline{x}_I)C(\underline{x};\underline{x}-\underline{x}_I) \tag{3}$$

where

$$C(\underline{x};\underline{x}-\underline{x}_I) = \underline{H}(\underline{x}-\underline{x}_I)^T\underline{b}(\underline{x}). \tag{4}$$

In the above, $\underline{H}(\underline{x}-\underline{x}_I)$ is a column vector consisting of complete n^{th} order monomials, and $\underline{b}(\underline{x})$ is a column vector of coefficients. The kernel function $\Phi_a(\underline{x}-\underline{x}_I)$ defines the locality of the approximation, and also the smoothness. For instance, a C^2 cubic B-spline kernel gives C^2 continuity of the approximation. In MEGA, various kernel functions with different levels of smoothness are available. The coefficients $\underline{b}(\underline{x})$ enforce the monomial reproducing conditions :

$$\sum_{I\in\mathcal{Z}} \Psi_I(\underline{x})\underline{H}(\underline{x}_I) = \underline{H}(\underline{x}). \tag{5}$$

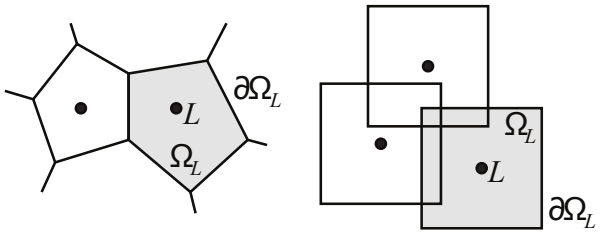
With the coefficients $\underline{b}(\underline{x})$ obtained from (4) and (5), the RK shape functions are constructed as :

$$\Psi_I(\underline{x}) = \underline{H}(\underline{0})\underline{M}(\underline{x})^{-1}\underline{H}(\underline{x}-\underline{x}_I)\Phi_a(\underline{x}-\underline{x}_I) \tag{6}$$

where

$$\underline{M}(\underline{x}) = \sum_{I\in\mathcal{Z}} \underline{H}(\underline{x}-\underline{x}_I)\underline{H}(\underline{x}-\underline{x}_I)^T\Phi_a(\underline{x}-\underline{x}_I). \tag{7}$$

In (7), the moment matrix needs to be invertible in order to be able to construct the RK shape functions (6). This requires a minimum number of nodes (with non-coplanar locations) covering a given point $\underline{x}$, which can be difficult to achieve in a simulation of extremely large deformations, particularly in fragment-impact problems. To remedy this situation, MEGA employs the quasi-linear RK approximation [10], which guarantees that the moment matrix is never singular.



(a) (b)

FIGURE 3 – Smoothing nodal domains for (a) SCNI and (b) SNNI.

3.3 Discretization

For discretization of (1), the displacements in (1) are approximated by (2). Employing the arbitrary nature of the virtual coefficients, we have the following semi-discrete matrix form :

$$\underline{\underline{M}}\,\underline{a} = \underline{f}^{ext} - \underline{f}^{int} \tag{8}$$

where $\underline{a}$ is the column vector of generalized accelerations $\{\underline{\ddot{d}}_I\}_{I\in\mathcal{Z}}$, $\underline{\underline{M}}$, $\underline{f}^{ext}$, and $\underline{f}^{int}$ are the mass matrix, external force, and internal force, respectively :

$$\underline{\underline{M}}_{IJ} = \underline{\underline{I}}_3 \int_{\Omega_{\underline{x}}} \Psi_I(\underline{x})\Psi_J(\underline{x})\rho d\underline{x} \tag{9a}$$

$$\underline{f}_I^{ext} = \int_{\Omega_{\underline{x}}} \Psi_I(\underline{x})\underline{b}(\underline{x})\rho d\underline{x} + \int_{\partial\Omega_{\underline{x}}^h} \Psi_I(\underline{x})\underline{t}(\underline{x}) d\underline{x} \tag{9b}$$

$$\underline{f}_I^{int} = \int_{\Omega_{\underline{x}}} \underline{\underline{B}}_I^T(\underline{x})\underline{\underline{\Sigma}}(\underline{x}) d\underline{x} \tag{9c}$$

where $\underline{\underline{I}}_3$ is a 3×3 identity matrix, $\underline{b}(\underline{x})$ and $\underline{t}(\underline{x})$ are the vector forms of the body force and prescribed tractions, respectively, $\underline{\underline{B}}(\underline{x})$ is the strain-gradient matrix, and $\underline{\underline{\Sigma}}(\underline{x})$ is the matrix form of the Cauchy stress.

4 Numerical Quadrature in MEGA

Nodal integration is the preferred method of quadrature in meshfree methods since it is numerically efficient and maintains desirable characteristics of meshfree methods on the discrete level. However, this approach can yield non-convergent results and solution instability [11] since the weak-form is under-integrated. Advanced nodal integration methods that address these issues are implemented in MEGA.

4.1 Stabilized Nodal Integration

A Stabilized Conforming Nodal Integration (SCNI) [11] has been introduced to remedy rank instability in direct nodal integration, and also provide optimal convergence for linearly complete shape functions. In this method, gradients are smoothed over conforming nodal representative domains which partition the domain as shown in Figure 3(a). Thus they are not evaluated directly at the nodes, avoiding rank instability. With the smoothed nodal gradients in hand, nodal integration is performed.

The smoothing is also done in such a way that the first order variational consistency condition (for Galerkin linear exactness) is ensured. This is the requirement to satisfy the following divergence equality with the set of test functions and the chosen numerical integration [11] :

$$\hat{\int}_{\Omega_{\underline{x}}} \underline{\nabla}_{\underline{x}}\Psi_I(\underline{x})\, \mathrm{d}\Omega = \hat{\int}_{\partial\Omega_{\underline{x}}} \Psi_I(\underline{x})\underline{n}(\underline{x})\, \mathrm{d}\Gamma \quad \forall I \tag{10}$$

where "$\hat{}$" denotes numerical integration, and $\Psi_I(\underline{x})$ is a shape function with first order completeness used in the Galerkin equation. SCNI considers gradient smoothing with divergence in each nodal representative domain by

$$\underline{\tilde{\nabla}}_{\underline{x}}\Psi_I(\underline{x}_L) = \frac{1}{|\Omega_L|}\int_{\Omega_L} \underline{\nabla}_{\underline{x}}\Psi_I(\underline{x})\, \mathrm{d}\Omega = \frac{1}{|\Omega_L|}\int_{\partial\Omega_L} \Psi_I(\underline{x})\underline{n}(\underline{x})\, \mathrm{d}\Gamma. \tag{11}$$

Here $|\Omega_L| = \int_{\Omega_L} d\Omega$ and Ω_L is the representative domain of node L. The conforming nodal domains can be generated by, e.g., Voronoi diagrams. When the gradient approximation (11) is employed in the Galerkin equation in conjunction with first-order complete approximations, the integration constraint (10) is satisfied, and exactness in linear problems (passing the linear patch test) is attained, along with optimal convergence rates associated with linear completeness.

The reformation of the conforming domains in SCNI such as those shown in Figure 3(a) can be cumbersome in the presence of extreme deformations, and stabilized non-conforming nodal integration (SNNI) [5] has been introduced as a simplification of SCNI. Figure 3(b) shows the gradient smoothing schemes by non-conforming cells constructed by considering box domains surrounding the node [12]. In MEGA, the smoothing cells are constructed to be conforming in the initial configuration.

The energy of sawtooth modes may still be under-sampled in smoothed nodal integration methods when the surface area-to-volume ratio of the domain is relatively small, or when the discretization is sufficiently fine. To this end, a Naturally Stabilized Nodal Integration (NSNI) [6] is implemented into MEGA under the strain-smoothing framework, which provides additional coercivity in nodal integration and precludes these sawtooth modes.

5 Constitutive modeling

The MEGA code implements small deformation elasticity and finite-strain plasticity models. The elastic model is only valid for small deformations and has not yet been generalized to large strains and rotations, e.g., viz the Neo-Hookean model. Plasticity models in MEGA include $J2$ (von Mises) plasticity with isotropic hardening and isotropic damage (e.g., for metals), and Drucker-Prager plasticity with tension cut-off and isotropic damage (e.g., for geomaterials).

5.1 Objective Stress update

MEGA employs the Jaumann rate of the Cauchy stress to maintain objectivity in constitutive modeling. For the time step from t^n to t^{n+1}, Hughes and Winget [13] showed that a hypo-elastic constitutive equation can be integrated in time in an *incrementally* objective way (preserving further objectivity in a discrete sense) by first defining an incremental deformation gradient $\underline{\underline{G}}$ with respect to the configuration $\underline{x}^{n+1/2} = 1/2\left(\underline{x}^n + \underline{x}^{n+1}\right)$:

$$G_{ij} = \frac{\partial \Delta d_i}{\partial x_j^{n+1/2}} \tag{12}$$

where $\Delta\underline{d} = \underline{d}^{n+1} - \underline{d}^n$ is the increment of displacement. In MEGA, the gradient (12) is computed using the smoothed shape functions in Sec. 4. The symmetric part of the gradient $\underline{\underline{G}}$ is then employed for the strain measurements in calculating the elastic trial stress in plasticity. With the trial stress in hand, the true Cauchy stress $\underline{\underline{\sigma}}^{n+1}$ at time $n+1$ is obtained via iteration on the plasticity equations. Then the internal force $\underline{f}_{int}^{n+1}$ is formed with nodal integration :

$$\underline{f}_{int}^{n+1} = \sum_{I \in \mathcal{Z}} \underline{\tilde{B}}^T(\underline{x}_I)\underline{\underline{\Sigma}}^{n+1}(\underline{x}_I)V_I \tag{13}$$

where $\underline{\tilde{B}}^T$ and $\underline{\underline{\Sigma}}^{n+1}$ are vectors containing the smoothed spatial gradients and stresses $\underline{\underline{\sigma}}^{n+1}$, respectively, and V_I is the nodal volume in the current configuration. The contribution to the internal force by NSNI follows analogously. Further details can be found in [6].

6 Time integration

Time integration on the semi-discrete form (8) is accomplished using the Newmark-β algorithm with the explicit central difference scheme ($\beta = 0$ and $\gamma = 1/2$). First, to make the algorithm fully explicit, the mass in (8) is lumped into a lumped-mass matrix $\underline{M}^l$ using the row-sum technique. In the explicit dynamic context, the Rayleigh damping is adopted with a purely mass-proportional damping matrix.

The predictor-corrector algorithm in *a-form* is employed; given the quantities at the previous timestep n, for time $n+1$, the predictors are first computed :

$$\underline{\widetilde{d}}_{n+1} = \underline{d}_n + \Delta t \underline{v}_n + \frac{\Delta t^2}{2}\underline{a}_n, \quad \underline{\widetilde{v}}_{n+1} = \underline{v}_n + \frac{\Delta t}{2}\underline{a}_n \tag{14}$$

Based on predictors, the shape functions, stresses, and internal forces are updated as described in Sections 3.2 and 5. With the mass lumping and mass-proportional damping in hand, the explicit central difference formulas yield the generalized accelerations at each node I :

$$\underline{a}_I^{n+1} = \frac{1}{M_{II}^l + \frac{1}{2}C_{II}\Delta t^2}\left\{ \left(\underline{f}_{ext}^{n+1}\right)_I - \left(\underline{f}_{int}^{n+1}\right)_I - C_{II}\left(\underline{\widetilde{v}}_{n+1}\right)_I \right\} \tag{15}$$

where M_{II}^l and C_{II} are the scalar mass and damping coefficient associated with node I from mass lumping and mass-proportional damping, respectively. The correctors are then calculated as

$$\underline{d}_{n+1} = \underline{\widetilde{d}}_{n+1}, \quad \underline{v}_{n+1} = \underline{\widetilde{v}}_{n+1} + \frac{\Delta t}{2}\underline{a}_{n+1} \tag{16}$$

which begins the process again with $n \leftarrow n+1$ in (15)-(16) for the next time step via the predictors (14).

7 Numerical Examples

7.1 Taylor bar impact problem

Consider an aluminum cylindrical bar impacting a rigid wall. This problem was experimentally performed in [14] and is often used to benchmark the implementation of a nonlinear large deformation computer code. The initial radius and height of the bar are 0.391 cm and 2.346 cm, respectively, and the impact velocity is 373 m/sec. $J2$ plasticity is employed to model the material; more details can be found in [3]. The numerical results and experimental data are compared in Table 1. For reference, a purely Lagrangian RKPM [15] solution is also computed using MEGA. As one can observe, the results from MEGA match well with the experimental data which indicates the success of the implementation for modeling nonlinear large deformation problems.

TABLE 1 – Comparison of deformed height and radius for the Talyor bar impact problem

	Lagrangian RKPM	Semi-Lagrangian RKPM	Experimental data [14]
Height (cm)	1.647	1.691	1.651
Radius (cm)	0.792	0.799	NA

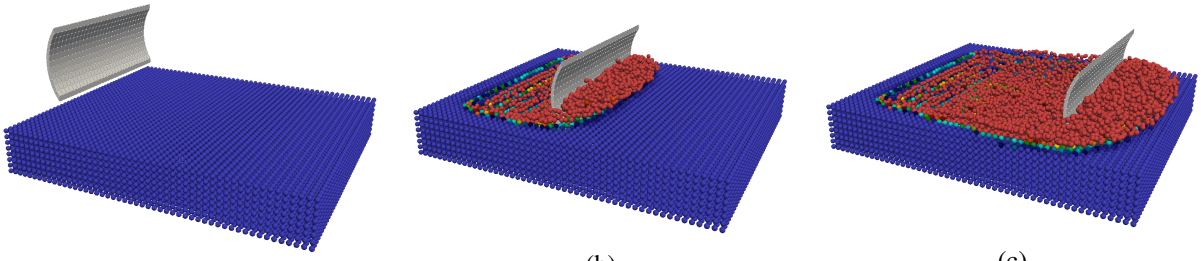
7.2 Earth Moving problem

For purely demonstrative purposes, an earth moving problem is simulated. A rigid blade digs into a soil bed and moves the material. The blade is modeled using RKPM with a prescribed displacement, while the soil is modeled using the Drucker-Prager material model. The progression of the simulation is shown in Figure 4.

8 Outlook

8.1 Current Applications and Limitations

The MEGA code is suitable for running many problems involving extremely large deformations. For instance, problems in geomechanics such as slope stability, landslide simulation, earth moving, tillage, and penetration and perforation.



(a) (b) (c)

FIGURE 4 – Earth moving simulation at a time (a) 0.0s, (b) 1.65s and (c) 3.3s

As an explicit dynamic code, the limitations of MEGA are inherited from this framework. Problems where the loadings and response are low frequency (e.g., structural dynamics) are not well-suited for analysis by MEGA, and should instead be solved using an implicit approach for a much more effective solution. The shared-memory implementation of MEGA is also limiting since large computing clusters currently cannot be utilized for calculations.

8.2 Future Directions

In the future, more material models will be added as needed (e.g., to simulate the deposition process in additive manufacturing). Frameworks will be developed in order to avoid constant recalculation of shape functions at each time step, which is a major cost of the calculations ; this is an open research topic. More accurate contact algorithms will also be implemented into the program. On the practical side, a distributed memory implementation using Message Passing Interface will be considered to accommodate different machine architectures, i.e., high-performance computing clusters, so that MEGA can solve problems with finer discretizations for more accurate solutions. Finally, work is already underway to convert the Fortran90 code to Julia, which can be leveraged for a robust CPU or GPU implementation.

Références

[1] Jiun-Shyan Chen, Michael Hillman, and Sheng-Wei Chi. Meshfree methods : progress made after 20 years. *Journal of Engineering Mechanics*, 143(4) :04017001, 2017.

[2] Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. Reproducing kernel particle methods. 20(8-9) :1081–1106, 1995.

[3] Jiun-Shyan Chen, Chunhui Pan, Cheng-Tang Wu, and Wing Kam Liu. Reproducing Kernel Particle Methods for large deformation analysis of non-linear structures. *Computer Methods in Applied Mechanics and Engineering*, 139(1-4) :195–227, 1996.

[4] KJ Bathe. *Finite element procedures*. Prentice-Hill, Englewoods Cliffs, 1982.

[5] Pai Chen Guan, Jiun-Shyan Chen, Y. Wu, H. Teng, J. Gaidos, K. Hofstetter, and M. Alsaleh. Semi-Lagrangian reproducing kernel formulation and application to modeling earth moving operations. *Mechanics of Materials*, 41(6) :670–683, 2009.

[6] Michael Hillman and Jiun-Shyan Chen. An accelerated, convergent, and stable nodal integration in Galerkin meshfree methods for linear and nonlinear mechanics. *International Journal for Numerical Methods in Engineering*, 107 :603–630, 2016.

[7] Nathan M Newmark. A method of computation for structural dynamics. *Journal of the engineering mechanics division*, 85(3) :67–94, 1959.

[8] Lisa S Avila, Sebastien Barre, Rusty Blue, Berk Geveci, Amy Henderson, William A Hoffman, Brad King, C Charles Law, Kenneth M Martin, and William J Schroeder. *The VTK User's Guide*. Kitware New York, 2010.

[9] Jiun-Shyan Chen and Hui-Ping Wang. New boundary condition treatments in meshfree computation of contact problems. *Computer Methods in Applied Mechanics and Engineering*, 187(3-4) :441–468, 2000.

[10] Edouard Yreux and Jiun-Shyan Chen. A quasi-linear reproducing kernel particle method. *International Journal for Numerical Methods in Engineering*, 109(7) :1045–1064, 2017.

[11] Jiun-Shyan Chen, Cheng-Tang Wu, and Sangpil Yoon. A stabilized conforming nodal integration for Galerkin mesh-free methods. *Int. J. Numer. Meth. Eng*, 0207(February 2000) :435–466, 2001.

[12] J.-S. Chen, Michael Hillman, and Marcus Rüter. An arbitrary order variationally consistent integration for Galerkin meshfree methods. *International Journal for Numerical Methods in Engineering*, 95(5) :387–418, 2013.

[13] Thomas JR Hughes and James Winget. Finite rotation effects in numerical integration of rate constitutive equations arising in large-deformation analysis. *International journal for numerical methods in engineering*, 15(12) :1862–1867, 1980.

[14] Mark L Wilkins and Michael W Guinan. Impact of cylinders on a rigid boundary. *Journal of Applied Physics*, 44(3) :1200–1206, 1973.

[15] Jiun-Shyan Chen, C Pan, CMOL Roque, and Hui-Ping Wang. A lagrangian reproducing kernel particle method for metal forming analysis. *Computational mechanics*, 22(3) :289–307, 1998.