Nodal density refinement and adaptivity for the meshless integral method

J. F. Ma, X. J. Xin, A. Bodin & P. Krishnaswami
Department of Mechanical and Nuclear Engineering, Kansas State University, USA

Abstract

The meshless integral method is an improved version from an earlier model based on the local boundary integral equation (LBIE). Through the use of the subtraction technique, the strong singularity in the governing integral equation is removed which significantly improves the accuracy and robustness of the method. The meshless integral method does not require a mesh to discretize the problem domain, and the approximate solution is constructed entirely based on a set of scattered nodes. Essential boundary conditions are imposed directly and accurately in a way similar to that used in FEM. In this paper, the effect of nodal density is investigated, and an adaptive algorithm is proposed for density refinement and accuracy improvement.

Keywords: meshless method, local boundary integral equation, linear elasticity, subtraction technique, singularity removal, nodal density refinement.

1 Introduction

The finite element method (FEM) has been one of the most widely used computational methods in mechanics. Despite its computational power and versatility, FEM remains prone to some inherent problems such as locking, limited continuity across element boundaries, difficulties in mesh generation, and slow convergence or even divergence in the event of severe mesh distortion or entanglement.

Over the past two decades the meshless methods in computational mechanics have attracted much attention owing to their advantage in adaptivity and in solving problems with discontinuity, large deformation, moving boundary, and material nonlinearity. In the meshless method, the concept of an element is
eliminated. The model geometry consists of a distribution of nodes over the domain and the approximate solution is constructed entirely based on a set of scattered nodes. Consequently, the nodal connectivity in the meshless method is much more flexible.

A meshless integral method by Bodin et al [1] based on the regularized integral equation is introduced and it is an improved method over Atluri et al [2]. It utilizes the boundary integral representation over a local contour and the moving least-squares approximation (MLSA) for function approximation. In this paper, the effect of nodal density is investigated, and an adaptive algorithm is proposed for density refinement and accuracy improvement.

2 Local boundary integral equation for linear elasticity

Consider an elastic body represented by a domain $\Omega$ with boundary $\Gamma$. The governing elasticity equations are as follows:

$$
\begin{align*}
\forall x \in \Omega, \quad & \sigma_{ij,j}(x) + b_i(x) = 0 \\
\epsilon_{ij}(x) = & \frac{1}{2}(u_{ij}(x) + u_{ji}(x)) \\
\sigma_{ij}(x) = & C_{ijkl} \varepsilon_{kl}(x)
\end{align*}
$$

(1)

where $\sigma$ and $\varepsilon$ are the stress and the strain tensor associated with the displacement $u$, $b$ is the body force in the domain $\Omega$, and $C$ is the elastic stiffness matrix. The essential and the natural boundary conditions on the boundary $\Gamma$ are defined respectively as $u_j = \bar{u}_j$ on $\Gamma_u$ and $t_j = \sigma_{ij} n_j = \bar{t}_j$ on $\Gamma_t$, with $\Gamma_t \cup \Gamma_u = \Gamma$ and $\Gamma_t \cap \Gamma_u = \emptyset$.

The weak form of (1) is:

$$
\int_{\Omega^{(c)}} (\sigma_{ij,j}(x) + b_i(x)) g_i(x,y^{(i)}) d\Omega(x) = 0
$$

(2)

where $\Omega^{<a>} \subset \Omega$ is a sub-domain related to node $y^{<a>}$, $x$ is the integration point, and $g_i$ is the test function.

In Bodin et al [1], special test functions $u_{ij}^*$ and $t_{ij}^*$ were used in eqn (2), and the subtraction technique was employed to remove strong singularity for boundary nodes. The following equation was obtained:

$$
\sigma_{ij}^{<a>} u_{ij}^{<a>} = \int_{\Gamma_u^{<a>}+\Gamma_t^{<a>}} \tilde{u}_{ij}^{<a>}(x,y^{<a>}) t_j(x) d\Gamma - \int_{C^{<a>}} t_{ij}^{<a>}(x,y^{<a>}) u_j(x) d\Gamma
\quad + \int_{\Gamma_u^{<a>}} t_{ij}^{<a>}(x,y^{<a>})(u_j(x) - u_j^{<a>}) d\Gamma
\quad + \int_{\Omega^{<a>}} b_j(x) \tilde{u}_{ij}^{<a>}(x,y^{<a>}) d\Omega
$$

(3)
where

\[
\[\alpha_{ij}^{(a)}\] = \begin{bmatrix}
\frac{\sin 2\theta_2 - \sin 2\theta_1}{2\pi} & \cos 2\theta_2 - \cos 2\theta_1 & \frac{\cos 2\theta_2 - \cos 2\theta_1}{2\pi (3 - 4\nu)} \\
\cos 2\theta_2 - \cos 2\theta_1 & \frac{\sin 2\theta_2 - \sin 2\theta_1}{2\pi} & \frac{\sin 2\theta_2 - \sin 2\theta_1}{2\pi (3 - 4\nu)} \\
\frac{\cos 2\theta_2 - \cos 2\theta_1}{2\pi (3 - 4\nu)} & \frac{\sin 2\theta_2 - \sin 2\theta_1}{2\pi (3 - 4\nu)} & \frac{\sin 2\theta_2 - \sin 2\theta_1}{2\pi (3 - 4\nu)} \\
\end{bmatrix}
\] (4)

Here \(\theta_2 - \theta_1 = \theta\) is the internal boundary angle subtended by material at \(y^{(a)}\) on the boundary.

For the meshless method, two useful terms are introduced: the sub-domain and the support domain. The sub-domain for node \(y^{(a)}\) is used for integration of the weak formulation of elasticity and is a sphere or part of a sphere centered on \(y^{(a)}\) with a given radius \(h_s^{(a)}\). The support domain for node \(y^{(a)}\) is used for moving least-squares approximation and is a sphere or part of a sphere centered on \(y^{(a)}\) with a given radius \(\ell_s^{(a)}\).

Two other frequently used terms are the domain of definition and the domain of influence. The domain of definition of a point \(x\) is the set of all nodes whose weight functions are non-zero at \(x\), while the domain of influence of a node \(y^{(a)}\) is the set of all nodes whose weight functions are non-zero in any part of the sub-domain of node \(y^{(a)}\).

We apply the MLSA to the eqn (3) to establish the meshless formulation. The shape function gives:

\[
u_j(x) = \sum_{b=1}^{N_x} \phi^{(b)}(x) u_j^{(b)} \quad \text{and} \quad u_{j,k}(x) = \sum_{b=1}^{N_x} \phi^{(b)}(x) u_{j,k}^{(b)},
\]

where \(N_x\) is the total number of nodes in the domain of definition of point \(x\).

The related traction term is \(t_j(x) = \sigma_{ij}(x)n_i(x)\), where \((n_1, n_2)\) is the normal to the plane passing through \(x\) over which the traction acts. For a node \(y^{(b)}\), we define \(N\) and \(B^{(b)}\) matrices as:

\[
N = \begin{bmatrix}
n_1 & 0 & n_2 \\
0 & n_2 & n_1
\end{bmatrix}; \quad B^{(b)} = \begin{bmatrix}
\phi_1^{(b)} & 0 \\
0 & \phi_2^{(b)} \\
\phi_2^{(b)} & \phi_1^{(b)}
\end{bmatrix}
\] (5)

We can then express the traction in terms of the shape functions as follows:

\[
\begin{bmatrix}
t_1(x) \\
t_2(x)
\end{bmatrix} = \sum_{b=1}^{N_x} \begin{bmatrix}
N \cdot CB^{(b)}(y^{(a)})
\end{bmatrix} \begin{bmatrix}
u_1^{(b)} \\
u_2^{(b)}
\end{bmatrix}
\] (6)

here \(C\) is the stiffness matrix.
With the above discretization and boundary conditions that \( u_j = \bar{u}_j \) on \( \Gamma^{<a>}_u \) and \( t_j = \bar{t}_j \) on \( \Gamma^{<a>}_t \), eqn (3) becomes:

\[
\sum_{b=1}^{N_y} \left( H_{ij}^{<a,b>} \hat{u}_j^{<b>} - \sum_{b=1}^{N_y} \left( L_{ij}^{<a,b>} \hat{u}_j^{<b>} \right) + \alpha^{<a>}_{ij} \right) u_j^{<a>} = G_i^{<a>}
\]

(7)

where \( H_{ij}^{<a,b>} \), \( L_{ij}^{<a,b>} \) and \( G_i^{<a>} \) are:

\[
H_{ij}^{<a,b>} = - \int_{\Gamma^{<a>}_t} \tilde{t}_{ij}^{*}(\mathbf{x}, y^{<a>}) \left( \varphi^{<b>}(\mathbf{x}) \right) d\Gamma - \int_{\Gamma^{<a>}_t} \tilde{t}_{ij}^{*}(\mathbf{x}, y^{<a>}) \left( \varphi^{<b>}(\mathbf{x}) \right) d\Gamma.
\]

(8)

\[
L_{ij}^{<a,b>} = \int_{\Gamma^{<a>}_t} \tilde{t}_{ij}^{*}(\mathbf{x}, y^{<a>}) \left( \varphi^{<b>}(\mathbf{x}) \right) d\Gamma.
\]

(9)

\[
G_i^{<a>} = \int_{\Omega^{<a>}} b_j(\mathbf{x}) \tilde{u}_{ij}^{*}(\mathbf{x}, y^{<a>}) d\Omega + \int_{\Gamma^{<a>}_t} \tilde{u}_{ij}^{*}(\mathbf{x}, y^{<a>}) \tilde{t}_j(\mathbf{x}) d\Gamma - \int_{\Gamma^{<a>}_u} \tilde{t}_{ij}^{*}(\mathbf{x}, y^{<a>}) (\bar{u}_j(\mathbf{x}) - \bar{u}_j(y^{<a>})) d\Gamma.
\]

(10)

The essential boundary conditions can be easily and directly imposed by replacing the governing equation corresponding to the DOF with prescribed displacement as follows:

\[
u_i^{<a>} = \sum_{b=1}^{N_y} \varphi^{<b>}(y^{<a>}) \hat{u}_j^{<b>} = \bar{u}_i^{<a>}
\]

(11)

Since \( t_i = \bar{t}_i \) has been incorporated in the calculation of \( G_i^{<a>} \) in eqn (10), no special treatment is needed for imposing the natural boundary conditions.

After the boundary conditions are imposed, the governing equations can be written as

\[
\sum_{b=1}^{N_y} \left( K_{ij}^{<a,b>} \hat{u}_j^{<b>} \right) = F_i^{<a>}
\]

(12)

where
\[ K_{ij}^{<a,b>} = \begin{cases} H_{ij}^{<a,b>} - L_{ij}^{<a,b>} + \alpha_{ij}^{<a>} \varphi_{<b>} (y^{<a>}) & \text{when } u_{i}^{<a>} \text{ is in constrained} \\ \varphi_{<b>} (y^{<a>}) \delta_{ij} & \text{when } u_{i}^{<a>} = \bar{u}_{i}^{<a>} \end{cases} \] 

(13)

and the upper limit of summation, \( N_y \), is the total number of nodes in the domain of influence of node \( y^{<a>} \).

### 3 Pre-processor for generating input file

A pre-processor has been developed for ease in preparing the input data file for the meshless analysis. To use the pre-processor, the user should provide three ASCII files:

The first file contains information about the problem domain, including the dimensionality, number of nodes, material properties, and the parameters required by the meshless method such as the order of monomial basis, the type of weight function, etc.

The second file contains the number of quadrilaterals that the domain is divided into for node generation, and the segments (LINE, ARC, and CIRCLE) used to define the quadrilaterals. It also gives the number of division for each segment with bias ratio that allows for the non-uniform division of a segment. Increasing the number of division refines the nodal density.

The third file specifies the boundary for the problem domain. It gives the number of boundaries, the number of segments in each boundary, and the type and geometry of each segment. The file also specifies the essential boundary conditions (1 means specified; 0 means unspecified).

For each quadrilateral provided in the second file, we use the quadrilateral-based method described in Algorithm 3.1 to generate the nodes. The algorithm is prescribed as follows:

**Algorithm 3.1: Generation of nodes within a given quadrilateral domain**

Given: Edges \( E(i), i = 0, 1, \ldots, 3 \) of the quadrilateral; type of each edge \( \text{Type}(j), j = 0, 1, \ldots, 3 \) (the value of each \( \text{Type}(j) \) can only be LINE or ARC); Start pt. \( (x_s(i), y_s(i)) \), end pt. \( (x_e(i), y_e(i)) \), center pt. \( (x_c(i), y_c(i)) \) for each edge \( i, i = 0, 1, \ldots, 3 \) (center pt. is ignored if it is a LINE); \( N_1 \), the number of divisions for 1st and 3rd edges, and \( R_1 \), the corresponding division ratio; \( N_2 \), the number of divisions for 2nd and 4th edges, and \( R_2 \), the corresponding division ratio.

Output: Locations \( \text{Loc}(N+k), k = 0, 1, \ldots, (N_1+1)*(N_2+1)-1 \) of the nodes generated, \( N \) is the initial size of \( \text{Loc} \).
Step 1: for (i = 0; i < 4; i++)

if (((i==0)||(i==2))  \{ q= e^{\frac{\ln R_i}{N_1}} ; N_0=N_1; \} \\
else \{q= e^{\frac{\ln R_i}{N_2}} ; N_0=N_2; \}
endif;

if (Type(i)==LINE); //Starting pt. (x(i), y(i)); ending pt. (x(e), y(e))

if(q==1) //uniform distribution of nodes

UnitX = (x(e)-x(i))/N0; UnitY=(y(e)-y(i))/N0;
Pts(i,N0) = (x(i), y(i));
Pts(i,j)=(x(i)+j*UnitX, y(i)+j*UnitY), j = 0, ..., N_0-1; 
//Pts are pts. on edge i.
else //non-uniform distribution of nodes

UnitX=(x(e)-x(i))*(1-q)/(1-qN0);UnitY=(y(e)-y(i))*(1-q)/(1-qN0)
Pts(i,0)= (x(i), y(i));
Pts(i,j)=(x(i)+UnitX*(1-qj)/(1-q),ys(i)+UnitY*(1-qj)/(1-q)),
j =1,..., N_0-1;
Pts(i,N0)= (xe(i), ye(i));
end if // end of if-else block on q
end if // end of if block for (Type(i)==LINE)

if(Type(i)==ARC) //Starting pt. (x(i), y(i)); ending pt. (x(e), y(e));

// center (x_c(i),y_c(i))

R=dist((x(i), y(i)), (x_c(i),y_c(i));
x_1=x(i)-x_c(i); y_1 =y(i)-y_c(i);//(x_1,y_1)vector of start pt. from center
x_2=x(e)-x(i); y_2 =y(e)-y(i);//(x_2,y_2)vector of end pt. from center
refX = xc(i)-x_1; refY=y_c-y_1;
\[ \alpha = \cos^{-1}\left(\frac{x_1 \ast x_2 + y_1 \ast y_2}{\sqrt{x_1^2 + y_1^2} \ast \sqrt{x_2^2 + y_2^2}}\right) \]; //dotproduct((x_1,y_1), (x_2,y_2))
\[ \alpha \_ref = \cos^{-1}\left(\frac{x_c(i)}{\sqrt{x_c(i)^2 + y_c(i)^2}}\right) \]; //dotproduct((x_1,y_1), (1,0))

if (crossproduct((x_2,y_2), (-x_1,-y_1))<0)  \alpha = 2 \ast \pi - \alpha ;
if (crossproduct((x_1,y_1),(-1,0))<0)  \alpha \_ref = 2 \ast \pi - \alpha \_ref ;
if (q==1) // uniform distribution of nodes

unitAngle=\frac{\alpha}{N_0};Pts(i,0) = (x(i), y(i));

Pts(i,j) = (x_c(i)+R*cos( \alpha \_ref +unitAngle*i),
y_c(i)+R*sin( \alpha \_ref +unitAngle*i)),j = 1, ..., N_0-1;
Pts(i,N0) = (x(i), y(i));
else //non-uniform distribution of nodes

unitAngle = \alpha \*(1-q)/(1- qN0);
Pts(i,0) = (x(i), y(i));
Pts(i,N0) = (x(i), y(i));
Pts(i, j) = (x(i)+R*cos( \alpha \_ref +unitAngle*(1- q')/(1-q)),

© 2005 WIT Press
WIT Transactions on Modelling and Simulation, Vol 41, © 2005 WIT Press
www.witpress.com, ISSN 1743-355X (on-line)
\[ y_c(i) + R \cdot \sin(\alpha_{\text{ref}} + \text{unitAngle} \cdot (1 - q) (1 - q)), j = 1, \ldots, N_0 - 1; \]
end if  // end of if-else block for q
end if  // end of if block for Type (i) == ARC
end if  // end of for loop on i

**Step 2:**
\[ L_1(i) = \text{Line}(\text{Pts}(0, i), \text{Pts}(2, i)), i = 0, 1, \ldots, N_1; \]
//Line from node i of E1 to node i of E3.
\[ L_2(j) = \text{Line}(\text{Pts}(1, j), \text{Pts}(3, j)), j = 0, 1, \ldots, N_2; \]
// similarly for E2 and E4.

**Step 3:**
\[ k = 0; \]
for (i=0; i<N_1+1; i++)
for (j=0; j<N_2+1; j++)
\[ \text{Loc}(N+k) = \text{Intersection}(L_1(i), L_2(j)); \]
\[ k = k + 1; \]
end for
end for

**Step 4:**
\[ N = N + (N_1 + 1) \cdot (N_2 + 1); \]
//update the size of the locations
// End of Algorithm 3.1

The set of nodes generated by Algorithm 3.1 will generally contain duplicates. This is because the nodes are generated quadrilateral by quadrilateral; thus, nodes that lie on an edge shared by two quadrilaterals are generated twice. The method that was developed for identifying and eliminating duplicates is detailed in Algorithm 3.2.

**Algorithm 3.2:** Eliminate duplicate nodes generated by Algorithm 3.1 and specify the type of each node

Given: \( \Omega = \{\text{Loc}(i), i=0, 1, \ldots, N-1\} \), the set of nodes generated by Algorithm 3.1 for all quadrilaterals of the problem domain; the global boundary segments \( \{S(k), k=0, 1, \ldots M-1\} \) where M is the number of segments; the start point \((x_s(k), y_s(k))\) and the end point \((x_e(k), y_e(k))\) of the k-th segment.

To find: the reduced set of nodes, \( \Omega_1 = \{\text{loc}(j), j=0, 1, \ldots, N_1-1\} \), after eliminating duplicates from \( \Omega \).

/*The initial size \( N_1 \) is 0, and the type of each node \text{type}(j)\), size is \( N_1 \) (if \text{loc}(j)\) is boundary node, \text{type}(j) = 1 otherwise \text{type}(j) = 0). If \text{loc}(j)\) is boundary node, also specify the two outward normals of this node: \text{leftNormal}(j), \text{rightNormal}(j). See Algorithm 3.3 for explanation of \text{leftNormal} and \text{rightNormal}. 

**getNormalSeg(k, (x, y))**: Return the outward normal of segment k at point (x, y).

*/

**Step 1:**
\[ N_1 = 0 \]
for (i=0; i<N; i++)
\[ \text{coincide} = \text{false}; \]
//flag for judging if this loc is already in \( \Omega_1 \)
for (k=0; k<N_1; k++)
if (\text{Loc}(i) == \text{loc}(k)) \{\text{coincide} = \text{true; break;};\}
end if
end for  // end of for loop on k
if (coincide==false) {loc(N1) = Loc(i); N1=N1+1;}
end if //end of if block on coincide
end for // end of for loop on i

Step 2: for (i=0; i<N1; i++)
flag = false; // judge if loc(i) is on boundary
for (j=0; j<M; j++)
if (loc(i) is on S(j));
flag=true;
if (loc(i) == (xs(j),ys(j)))
leftNormal(i) = getNormalSeg(j,(xs(j),ys(j)));
else if (loc(i) == (xe(j),ye(j))) //if loc(i) is ending pt. of S(j)
rightNormal(i) = getNormalSeg(j, (xe(j),ye(j));
else // smooth boundary location
leftNormal(i) = getNormalSeg(j, loc(i));
rightNormal(i) = getNormalSeg(j, loc(i));
end if // end of if-else block for the relationship between node
//and boundary.
end if
end for // end of for loop for j
if (flag==true) type(i) = 1;
else type(i) = 0;
end if // end of if-else block for flag
end for // end of for loop for i
// End of Algorithm 3.2

The given nodal information and the computed information (node locations, sub-domain radius, support domain radius, and essential boundary conditions) are then written to a data file in ASCII format. This data file will be read in by the meshless solver.

4 Numerical results

We used the global L2-norm error as a measure of the overall performance of the meshless method. The meshless integral method successfully passes constant stress patch tests with virtually machine accuracy and higher order patch tests. For a cantilever beam problem, the method also works well Bodin et al [1].

Here, the test example of an infinite plate with a circular hole subjected to a uniform remote tension $\sigma$ in $x_1$ direction is presented with details. The exact solutions for the stresses are given by Timoshenko and Goodier [3].

Because of symmetry, only the upper right quadrant of the plate was modeled. The model geometry has 336 nodes with more nodes concentrated around the hole. The material properties used in the calculation are $E=1000$ MPA, $\nu=0.3$ with plane strain condition, and $\sigma=1$ MPA. The essential boundary conditions were applied on the left and the bottom edges. On the remaining edges, the natural boundary conditions (according to analytical solution for an infinite plate with a hole) were applied.
The pre-processor described in Section 3 was used to generate the input files, and meshless models with various nodal density were tested: a coarse model with 336 nodes, a medium model with 1271 nodes, and a fine model with 2806 nodes. The linear monomial basis and spline weight function were used in each case. The L2-norm errors in displacement for these three cases are 0.0033, 0.0023, and 0.0018 respectively. Figure 1 shows the comparison.

![Figure 1](image)

**Figure 1:** L2-norm error in displacement for different nodal density using linear basis.

For comparison, we also used quadratic monomial basis and spline weight function to test the effect of nodal density. Figure 2 shows the comparison. From figure 1 and figure 2 we conclude that if the monomial basis is linear, the meshless method solution converges to the actual displacement field as the nodal distribution is refined.

![Figure 2](image)

**Figure 2:** L2-norm error in displacement for different nodal density using quadratic basis.
5 Conclusions

In this paper, a meshless integral method based on regularized local boundary integral equation is introduced. In the meshless integral method, the model geometry is made of a distribution of nodes over the domain and the approximate solution is constructed entirely based on a set of scattered nodes. An efficient pre-processor is presented for defining the data and discretization scheme for meshless analysis which can significantly reduces the manpower required for node generation. An infinite plate with a circular hole in the center is used to test the capacity of the pre-processor and the effect of nodal density is investigated.

References

