CHAPTER 16

MESHFREE METHOD AND APPLICATION TO SHAPE OPTIMIZATION

J. S. Chen

Civil & Environmental Engineering Department
University of California, Los Angeles
Los Angeles, CA 90095-1593
E-mail: jschen@seas.ucla.edu

Nam Ho Kim

Mechanical & Aerospace Engineering Department
University of Florida
Gainesville, Florida 32611-6250
E-mail: nkim@ufl.edu

Recent developments in meshfree method and its application to shape optimization are presented. The approximation theory of the Reproducing Kernel Particle Method is first introduced. The computational issues in domain integration and imposition of boundary conditions are discussed. A stabilization of nodal integration in meshfree discretization of boundary value problems is presented. Shape optimization based on meshfree method is presented, and the treatment of essential boundary conditions as well as the dependence of the shape function on the design variation is discussed. The proposed meshfree based shape design optimization yields a significantly reduced number of design iterations due to the meshfree approximation of sensitivity information without the need of remeshing. It is shown through numerical examples that the mesh distortion difficulty exists in the finite element–based design approach for design problems with large shape changes is effectively resolved.

1. Introduction

Meshfree methods developed in recent years introduced new approximation methods that are less restrictive in meeting the regularity requirement in the approximation and discretization of partial differential equations.[1-10] These methods are more flexible in embedding special enrichment functions in the approximation for solving problems with known characteristics, such as fracture problems,[11] more straightforward in constructing h– or p–adaptive refinement,[12-14]

and less sensitive to large geometry changes such as those in large deformation problems[5,15] and shape optimization problems.[16,17]

Primary computational challenges involved in shape optimization using finite element methods (FEM) arise from the excessive mesh distortion that occurs during large shape changes and mesh–dependent solution accuracy. Numerous difficulties were encountered in finite element analysis, such as those involving mesh distortion, mesh adaptation, and those with the need for a large number of re-meshing during shape optimization.[18,19] Meshfree method is ideal for shape optimization because it allows field variables to be interpolated at the global level, therefore avoiding the use of a mesh. The main purpose of this chapter is to introduce special features of the meshfree method from a design sensitivity analysis (DSA) and optimization viewpoint, as well as the associated numerical aspects. Mesh distortion and re-meshing problems encountered in FEM-based shape optimization can be avoided and the design costs can be significantly reduced as a result of the accurate and efficient computation of design sensitivity. An important aspect to be considered in the shape optimization using meshfree method is the design derivation of meshfree shape functions. Unlike the finite element shape functions which are independent to the design variation due to the local construction of shape functions using natural coordinates, the meshfree shape functions depend on a global coordinate of material points that are related to the design parameters in shape DSA. Thus, the design derivative of the meshfree shape functions needs to be considered in DSA.

This Chapter is organized as follows. In section 2, the reproducing kernel approximation for solving boundary value problems under the framework of reproducing kernel particle method (RKPM) is first introduced. Methods to impose boundary conditions and issues associated with the domain integration of Galerkin approximation are discussed, and an example demonstrating the accuracy and convergence property of RKPM is presented. In section 3, shape design parameterization and design velocity are first defined. The shape sensitivity derivation is then introduced, and design derivation of meshfree shape functions and RKPM discretization of sensitivity equation are discussed. Two shape design optimization problems solved using the proposed methods are presented in Section 4. Concluding remarks are given in Section 5.

2. Reproducing Kernel Particle Method

In meshfree methods, the approximation of unknowns in the partial differential equations are constructed entirely based on a set of discrete points without using a structured mesh topology. Approximation methods such as moving least-

squares,[20] reproducing kernel approximation,[4] partition of unity,[7] radial basis functions,[21] among others, have been introduced in formulating meshfree discrete equations. For demonstration purposes, the reproducing kernel approximation is presented herein. Other methods can also be employed under this framework.

2.1 Reproducing Kernel Approximation

The reproducing kernel approximation of a function $u(\mathbf{x})$, denoted by $u^h(\mathbf{x})$, is expressed as

$$u^h(\mathbf{x}) = \sum_{I=1}^{NP} \Psi_I(\mathbf{x}) d_I \,, \tag{1}$$

where NP is the number of points used in the discretization of the problem domain Ω, d_I is the coefficient of the approximation at point I, and $\Psi_I(\mathbf{x})$ is called the reproducing kernel shape function. The reproducing kernel shape function is formed by a multiplication of two functions

$$\Psi_I(\mathbf{x}) = C(\mathbf{x};\mathbf{x}-\mathbf{x}_I)\Phi_a(\mathbf{x}-\mathbf{x}_I)\,, \tag{2}$$

where $\Phi_a(\mathbf{x}-\mathbf{x}_I)$ is a kernel function that defines the continuity (smoothness) and the locality of the approximation with compact support (cover) Ω_I measured by the parameter a. The order of continuity in this approximation can be introduced without complexity. For example, the box function gives C^{-1} continuity, the hat function leads to C^0 continuity, the quadratic spline function results in C^1 continuity, and the cubic B-spline function yields C^2 continuity, etc. A commonly used kernel function is the cubic B-spline function given as

$$\Phi_a(x-x_I) = \begin{cases} \frac{2}{3} - 4z^2 + 4z^3 & for\ z \le \frac{1}{2} \\ \frac{4}{3} - 4z + 4z^2 - \frac{4}{3}z^3 & for\ \frac{1}{2} < z \le 1 \\ 0 & for\ z > 1 \end{cases}, \tag{3}$$

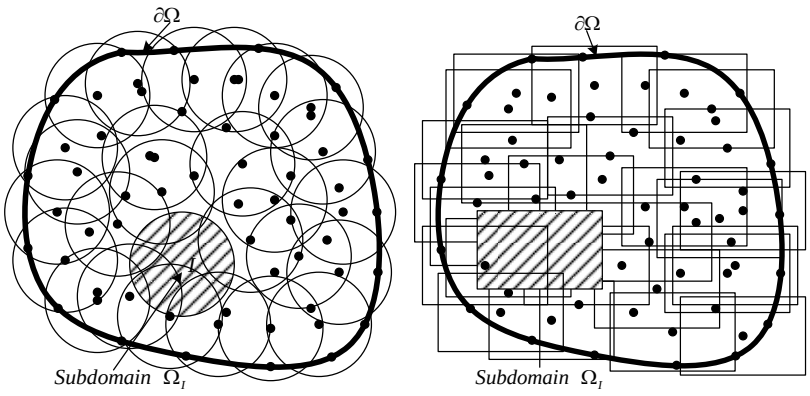
where $z = |x - x_I|/a$. In multi-dimension, the kernel function can be constructed by using the distance function to yield an isotropic kernel function,

$$\Phi_a(\mathbf{x}-\mathbf{x}_I) = \Phi_a(z), \quad z = \|\mathbf{x}-\mathbf{x}_I\|/a\,, \tag{4}$$

or by the tensor product of the one-dimensional kernel functions to yield a anisotropic kernel

$$\Phi_a(\mathbf{x}-\mathbf{x}_I) = \Phi_{a_1}(x_1 - x_{1I})\Phi_{a_2}(x_2 - x_{2I})\,. \tag{5}$$

The union of all the kernel supports (covers) should cover the entire problem domain, i.e., $\bigcup_I \Omega_I \supset \Omega$ as shown in Figure 1. The term $C(\mathbf{x},\mathbf{x}-\mathbf{x}_I)$ in Eq. (2) is the correction function or enrichment function. This function determines the completeness of the approximation and the order of consistency in solving PDE's.



(a) Isotropic kernel supports (covers) (b) Anisotropic kernel supports (covers)
Figure 1 Domain discretization and kernel supports

In general, $C(\mathbf{x},\mathbf{x}-\mathbf{x}_I)$ is constructed by the monomial bases:

$$\begin{aligned} C(\mathbf{x};\mathbf{x}-\mathbf{x}_I) &= \sum_{i+j=0}^{n} (x_1-x_{1I})^i (x_2-x_{2I})^j b_{ij}(\mathbf{x}) \\ &= \mathbf{H}^T(\mathbf{x}-\mathbf{x}_I)\mathbf{b}(\mathbf{x}), \qquad\qquad i,j \geq 0 \end{aligned} \tag{6}$$

$$\mathbf{H}^T(\mathbf{x}-\mathbf{x}_I) = \begin{bmatrix} 1 & x_1-x_{1I} & x_2-x_{2I} & \cdots & (x_2-x_{2I})^n \end{bmatrix} \tag{7}$$

$$\mathbf{b}(\mathbf{x}) = \begin{bmatrix} b_{00} & b_{10} & b_{01} & \cdots & b_{0n} \end{bmatrix}, \tag{8}$$

where n is the order of the complete monomial bases, and this number defines the completeness of the approximation. The unknown coefficients $\mathbf{b}(\mathbf{x})$ can be determined by imposing the *n*-th order reproducing conditions

$$\sum_{I=1}^{NP} \Psi_I(\mathbf{x}) x_{1I}^i x_{2I}^j = x_1^i x_2^j \quad i+j = 0,\cdots,n\,, \tag{9}$$

or equivalently in the following form:

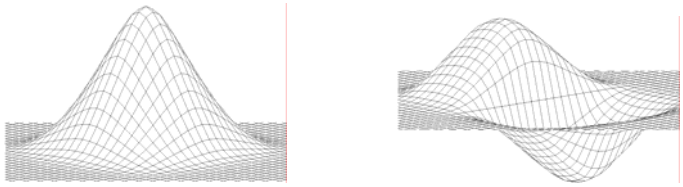
$$\sum_{I=1}^{NP} \Psi_I(\mathbf{x})(x_1-x_{1I})^i (x_2-x_{2I})^j = \delta_{i0}\delta_{j0} \quad i+j = 0,\cdots,n\,. \tag{10}$$

Substituting Eqs. (6) and (2) into Eq. (10) results in

$$\mathbf{M}(\mathbf{x})\mathbf{b}(\mathbf{x}) = \mathbf{H}(\mathbf{0})\,, \tag{11}$$

where $\mathbf{M}(\mathbf{x})$ is the moment matrix of the kernel function $\Phi_a(\mathbf{x}-\mathbf{x}_I)$

$$\mathbf{M}(\mathbf{x}) = \sum_{I=1}^{NP} \mathbf{H}(\mathbf{x}-\mathbf{x}_I)\mathbf{H}^T(\mathbf{x}-\mathbf{x}_I)\Phi_a(\mathbf{x}-\mathbf{x}_I)\,. \tag{12}$$



(a) (b)

Figure 2. (a) 2-dimensional meshfree shape function Ψ_I (b) Shape function derivative $\Psi_{I,x}$

Therefore, the coefficient function $\mathbf{b}(\mathbf{x})$ is solved by

$$\mathbf{b}(\mathbf{x}) = \mathbf{M}^{-1}(\mathbf{x})\mathbf{H}(\mathbf{0}). \tag{13}$$

Notice that for the moment matrix $\mathbf{M}(\mathbf{x})$ to be non-singular, any spatial position $\mathbf{x}$ should be covered at least by *n* linearly independent kernel functions such that the *n* reproducing equations are solvable. Subsequently, the correction function and the discrete reproducing kernel shape function can be obtained as

$$C(\mathbf{x};\mathbf{x}-\mathbf{x}_I) = \mathbf{H}^T(\mathbf{0})\mathbf{M}^{-1}(\mathbf{x})\mathbf{H}(\mathbf{x}-\mathbf{x}_I) \tag{14}$$

$$\Psi_I(\mathbf{x}) = \mathbf{H}^T(\mathbf{0})\mathbf{M}^{-1}(\mathbf{x})\mathbf{H}(\mathbf{x}-\mathbf{x}_I)\Phi_a(\mathbf{x}-\mathbf{x}_I). \tag{15}$$

The plots of the shape function and its derivatives are given in Figure 2, where linear basis and cubic B-spline kernel functions are employed. The meshfree shape function $\Psi_I(\mathbf{x})$ does not possess the Kronecker delta property; therefore, additional treatments are required to enforce the essential boundary conditions.

2.2 Galerkin Approximation and Discretization

For demonstration purposes, consider the following elastostatic problem:

$$(\sigma_{ij})_{,j} + b_i = 0 \quad in \quad \Omega \tag{16}$$

$$u_i = g_i \quad on \quad \partial\Omega^g \tag{17}$$

$$\sigma_{ij}n_j = h_i \quad on \quad \partial\Omega^h \tag{18}$$

where $\sigma_{ij} = C_{ijkl}\varepsilon_{kl}$, $\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \equiv (\nabla^S u)_{ij}$ in Ω, Ω is the problem domain with essential boundary $\partial\Omega^g$ and natural boundary $\partial\Omega^h$, u_i is displacement, σ_{ij} is stress, ε_{ij} is strain, C_{ijkl} is the elasticity tensor, h_i is the surface traction, and b_i is the body force. The weak form of the above problem is

$$\int_\Omega \delta(\nabla^S u)_{ij} C_{ijkl}(\nabla^S u)_{kl}\, d\Omega - \int_\Omega \delta u_i b_i d\Omega - \int_{\partial\Omega^h} \delta u_i h_i d\Gamma - \delta \int_{\partial\Omega^g} \lambda_i(u_i - g_i) d\Gamma = 0 \tag{19}$$

where λ_i is the Lagrange multiplier to impose the essential boundary conditions. By employing the displacement approximation in Eq. (1), and introducing approximation function $\{\phi_I(\mathbf{x})\}_{I=1}^{Ng}$ for the approximation of λ_i on $\partial\Omega^g$, where Ng is number of points on $\partial\Omega^g$, the following discrete equation is obtained

$$\begin{bmatrix} \mathbf{K} & \mathbf{G}^{\mathrm{T}} \\ \mathbf{G} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{d} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{q} \end{bmatrix} \tag{20}$$

$$\mathbf{K}_{IJ} = \int_{\Omega} \mathbf{B}_I^T \mathbf{C} \mathbf{B}_J d\Omega\,,\ \mathbf{G}_{IJ} = \mathbf{I} \int_{\partial\Omega^g} \phi_I \Psi_J d\Gamma\,,\ \mathbf{f}_I = \int_{\Omega} \Psi_I \mathbf{b} d\Omega + \int_{\partial\Omega^h} \Psi_I \mathbf{h} d\Gamma\,,$$

$$\mathbf{q}_I = \int_{\partial\Omega^g} \phi_I \mathbf{g} d\Gamma\,,\ \mathbf{B}_I = \begin{bmatrix} \Psi_{I,1} & 0 \\ 0 & \Psi_{I,2} \\ \Psi_{I,2} & \Psi_{I,1} \end{bmatrix} \tag{21}$$

Compared with the standard finite element Galerkin approximation, two major disadvantages in computational efficiency are observed. First, additional degrees of freedom for Lagrange multiplier for imposition of essential boundary conditions are needed. Second, domain integration requires a background grid if Gauss integration is to be employed. It has been observed by Belytschko *et al.*[22] that very high order quadrature rule and very fine integration grids are required to reduce the integration error, and thus yields poor efficiency.

2.3 Two Alternative Methods for Imposition of Essential Boundary Conditions

2.3.1 Transformation Method

Recall reproducing kernel approximation of displacement

$$u_i^h(\mathbf{x}) = \sum_{I=1}^{NP} \Psi_I(\mathbf{x}) d_{iI}\,. \tag{22}$$

The nodal value of u_i^h at node *J*, $\hat{d}_{iJ}$, is obtained by

$$\hat{d}_{iJ} = u_i^h(\mathbf{x}_J) = \sum_{I=1}^{NP} \Psi_I(\mathbf{x}_J) d_{iI}\,, \text{ or } \hat{\mathbf{d}} = \boldsymbol{\Lambda}\mathbf{d}\,, \tag{23}$$

where

$$\hat{\mathbf{d}}_I = \begin{bmatrix} \hat{d}_{1I} \\ \hat{d}_{2I} \end{bmatrix},\quad \mathbf{d}_I = \begin{bmatrix} d_{1I} \\ d_{2I} \end{bmatrix},\quad \boldsymbol{\Lambda}_{IJ} = \begin{bmatrix} \Psi_J(\mathbf{x}_I) & 0 \\ 0 & \Psi_J(\mathbf{x}_I) \end{bmatrix} = \Psi_J(\mathbf{x}_I)\mathbf{I} \tag{24}$$

and $\boldsymbol{\Lambda}$ is the transformation matrix between generalized displacement vector $\mathbf{d}$ and nodal displacement vector $\hat{\mathbf{d}}$.

For computational efficiency, only the degrees of freedom associated with the essential boundaries are expressed in the nodal coordinate. Following Chen and Wang[23], the discrete points are first partitioned into two groups: a boundary group G^B containing all points on $\partial\Omega^g$, and an interior group G^I containing the rest of the points. Further partitioning the displacement vectors into boundary and interior components, $\mathbf{d}^T = \begin{bmatrix}\mathbf{d}^{B^T} & \mathbf{d}^{I^T}\end{bmatrix}$, $\hat{\mathbf{d}}^T = \begin{bmatrix}\mathbf{d}^{B^T} & \mathbf{d}^{I^T}\end{bmatrix}$, we rewrite Eq. (23) as:

$$\hat{\mathbf{d}} = \begin{bmatrix}\hat{\mathbf{d}}^B \\ \hat{\mathbf{d}}^I\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Lambda}^{BB} & \boldsymbol{\Lambda}^{BI} \\ \boldsymbol{\Lambda}^{IB} & \boldsymbol{\Lambda}^{II}\end{bmatrix}\begin{bmatrix}\mathbf{d}^B \\ \mathbf{d}^I\end{bmatrix} \equiv \boldsymbol{\Lambda}\mathbf{d}\,. \tag{25}$$

Next, define a mixed displacement vector $\boldsymbol{d}^*$

$$\mathbf{d}^* = \begin{bmatrix}\hat{\mathbf{d}}^B \\ \mathbf{d}^I\end{bmatrix} = \begin{bmatrix}\boldsymbol{\Lambda}^{BB} & \boldsymbol{\Lambda}^{BI} \\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{d}^B \\ \mathbf{d}^I\end{bmatrix} = \boldsymbol{\Lambda}^*\mathbf{d} \tag{26}$$

or

$$\mathbf{d} = \boldsymbol{\Lambda}^{*-1}\mathbf{d}^*, \quad \boldsymbol{\Lambda}^{*-1} = \begin{bmatrix}\boldsymbol{\Lambda}^{BB-1} & -\boldsymbol{\Lambda}^{BB-1}\boldsymbol{\Lambda}^{BI} \\ \boldsymbol{0} & \boldsymbol{I}\end{bmatrix}. \tag{27}$$

The displacement approximation can now be approximated as

$$\mathbf{u}^h(\mathbf{x}) = \boldsymbol{\Psi}(\mathbf{x})\mathbf{d} = \boldsymbol{\Psi}(\mathbf{x})\boldsymbol{\Lambda}^{*-1}\mathbf{d}^* \quad , \tag{28}$$

where

$$\boldsymbol{\Psi}(\mathbf{x}) = \begin{bmatrix}\boldsymbol{\Psi}_1(\mathbf{x}) & \boldsymbol{\Psi}_2(\mathbf{x}) & \cdots & \boldsymbol{\Psi}_{NP}(\mathbf{x})\end{bmatrix}, \quad \boldsymbol{\Psi}_I(\mathbf{x}) = \begin{bmatrix}\Psi_I(\mathbf{x}) & 0 \\ 0 & \Psi_I(\mathbf{x})\end{bmatrix}. \tag{29}$$

With Eq. (28), kinematically admissible approximation of $u_i^h \in H_g^1$ and $\delta u_i^h \in H_0^1$ can be constructed, and the Galerkin approximation of weak form can be stated as: Find $u_i^h \in H_g^1$, $\forall \delta u_i^h \in H_0^1$, such that the following equation is satisfied:

$$\int_\Omega \delta(\nabla^S u^h)_{ij} C_{ijkl} (\nabla^S u^h)_{kl}\, d\Omega - \int_\Omega \delta u_i^h b_i\, d\Omega - \int_{\partial\Omega^h} \delta u_i^h h_i\, d\Gamma = 0\,. \tag{30}$$

Consider taking $\delta\mathbf{u}^h(\mathbf{x}) = \boldsymbol{\Psi}(\mathbf{x})\boldsymbol{\Lambda}^{*-1}\delta\mathbf{d}^*$ and $\mathbf{u}^h(\mathbf{x}) = \boldsymbol{\Psi}(\mathbf{x})\mathbf{d}$ in Eq. (30) to yield

$$\delta\mathbf{d}^{*T}\boldsymbol{\Lambda}^{*-T}\mathbf{K}\mathbf{d} = \delta\mathbf{d}^{*T}\boldsymbol{\Lambda}^{*-T}\mathbf{f} \quad \text{or} \quad \delta\mathbf{d}^{*T}\left(\mathbf{K}^*\mathbf{d} - \mathbf{f}^*\right) = 0\,, \tag{31}$$

where

$$\mathbf{K}^* = \boldsymbol{\Lambda}^{*-T}\mathbf{K} = \begin{bmatrix}\mathbf{K}^{*BB} & \mathbf{K}^{*BI} \\ \mathbf{K}^{*IB} & \mathbf{K}^{*II}\end{bmatrix}, \quad \mathbf{f}^* = \boldsymbol{\Lambda}^{*-T}\mathbf{f} = \begin{bmatrix}\mathbf{f}^{*B} \\ \mathbf{f}^{*I}\end{bmatrix}. \tag{32}$$

Let *Nb* be the total number of degrees of freedom associated with essential boundary conditions on $\partial\Omega^g$. By considering the essential boundary conditions, we have $\hat{\mathbf{d}}^B = \mathbf{g}$, $\delta\hat{\mathbf{d}}^B = \mathbf{0}$, and $\delta\mathbf{d}^{*T} = \begin{bmatrix}\mathbf{0}^T & \mathbf{d}^{IT}\end{bmatrix}$. By examining the discrete weak form, $\delta\mathbf{d}^{*T}\left(\mathbf{K}^*\mathbf{d} - \mathbf{f}^*\right) = 0$, it is apparent that the first *Nb* equations of $\mathbf{K}^*\mathbf{d} - \mathbf{f}^* = \mathbf{0}$ become redundant and can be replaced by the *Nb* equations of essential boundary conditions $\begin{bmatrix}\mathbf{\Lambda}^{BB} & \mathbf{\Lambda}^{BI}\end{bmatrix}\mathbf{d} = \mathbf{g}$ (Eq. (26)) to yield

$$\begin{bmatrix}\mathbf{\Lambda}^{BB} & \mathbf{\Lambda}^{BI} \\ \mathbf{K}^{*IB} & \mathbf{K}^{*II}\end{bmatrix}\begin{bmatrix}\mathbf{d}^B \\ \mathbf{d}^I\end{bmatrix} = \begin{bmatrix}\mathbf{g} \\ \mathbf{f}^{*I}\end{bmatrix}. \tag{33}$$

2.3.2 Modified Reproducing Kernel with Nodal Interpolation Properties

In this section we introduce an alternative method to construct a reproducing kernel approximation with Kronecker delta properties so that the essential boundary conditions can be imposed directly. Chen *et al.*[24] proposed a general formulation for developing reproducing kernel approximation with nodal interpolation properties.

Consider a modified reproducing kernel approximation of $u(\boldsymbol{x})$ as follows:

$$u_i^h(\mathbf{x}) = \sum_I \Psi_I(\mathbf{x})d_{iI} = \sum_I \left(\hat{\Psi}_I(\mathbf{x}) + \bar{\Psi}_I(\mathbf{x})\right)d_{iI}\,. \tag{34}$$

In Eq. (34), $\hat{\Psi}_I(\mathbf{x})$ is a primitive function used to introduce discrete Kronecker delta properties, and $\bar{\Psi}_I(\mathbf{x})$ is an enrichment function for imposing n-th order reproducing conditions. Consider the following construction of $\hat{\Psi}_I(\mathbf{x})$:

$$\hat{\Psi}_I(\mathbf{x}) = \frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x} - \mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})}, \qquad \hat{a}_I < \min\{\|\mathbf{x}_I - \mathbf{x}_J\|, \forall J \neq I\}\,. \tag{35}$$

The support size $\hat{a}_I$ of $\hat{\Phi}_{\hat{a}_I}(\mathbf{x} - \mathbf{x}_I)$ is so chosen that it does not cover any neighbor points, and thus Kronecker delta conditions are satisfied in $\hat{\Psi}_I(\mathbf{x})$. The enrichment function is taken as the standard reproducing kernel form as

$$\bar{\Psi}_I(\mathbf{x}) = \mathbf{H}^T(\mathbf{x} - \mathbf{x}_I)\mathbf{a}(\mathbf{x})\bar{\Phi}_{\bar{a}_I}(\mathbf{x} - \mathbf{x}_I)\,. \tag{36}$$

The coefficients $\mathbf{a}(\mathbf{x})$ in $\bar{\Psi}_I(\mathbf{x})$ are obtained by the following reproducing conditions:

$$\sum_I [\frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x} - \mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} + \mathbf{H}^T(\mathbf{x} - \mathbf{x}_I)\mathbf{a}(\mathbf{x})\bar{\Phi}_{\bar{a}_I}(\mathbf{x} - \mathbf{x}_I)]x_{1I}^i x_{2I}^j = x_1^i x_2^j, \quad 0 \leq i + j \leq n\,. \tag{37}$$

Equation (37) can be rewritten as:

$$\sum_I [\frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} + \mathbf{H}^T(\mathbf{x}\text{ - }\mathbf{x}_I)\mathbf{a}(\mathbf{x})\overline{\Phi}_{\overline{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)(x_1\text{ - }x_{1I})^i(x_2\text{ - }x_{2I})^j] = \delta_{0i}\delta_{0j}, \quad 0 \le i+j \le n \tag{38}$$

or

$$\sum_I [\frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} + \mathbf{H}^T(\mathbf{x}\text{ - }\mathbf{x}_I)\mathbf{a}(\mathbf{x})\overline{\Phi}_{\overline{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)\mathbf{H}(\mathbf{x}\text{ - }\mathbf{x}_I)] = \mathbf{H}(\mathbf{0})\,. \tag{39}$$

The coefficient vector $\mathbf{a}(\mathbf{x})$ is obtained from Eq. (39) by

$$\mathbf{a}(\mathbf{x}) = \mathbf{Q}^{-1}(\mathbf{x})[\mathbf{H}(\mathbf{0})\text{ - }\hat{\mathbf{F}}(\mathbf{x})] \tag{40}$$

$$\hat{\mathbf{F}}(\mathbf{x}) = \sum_I \mathbf{H}(\mathbf{x}\text{ - }\mathbf{x}_I)\frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} \tag{41}$$

Finally, the reproducing kernel interpolation function is obtained:

$$\Psi_I(\mathbf{x}) = \frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} + \mathbf{H}^T(\mathbf{x}\text{ - }\mathbf{x}_I)\mathbf{Q}^{-1}(\mathbf{x})[\mathbf{H}(\mathbf{0})\text{ - }\hat{\mathbf{F}}(\mathbf{x})]\overline{\Phi}_{\overline{a}_I}(\mathbf{x}\text{ - }\mathbf{x}_I) \tag{42}$$

The Kronecker delta properties can be easily shown:

$$\begin{aligned}\Psi_I(\mathbf{x}_J) &= \frac{\hat{\Phi}_{\hat{a}_I}(\mathbf{x}_J\text{ - }\mathbf{x}_I)}{\hat{\Phi}_{\hat{a}_I}(\mathbf{0})} + \mathbf{H}^T(\mathbf{x}_J\text{ - }\mathbf{x}_I)\mathbf{Q}^{-1}(\mathbf{x}_J)[\mathbf{H}(\mathbf{0})\text{ - }\hat{\mathbf{F}}(\mathbf{x}_J)]\overline{\Phi}_{\overline{a}_I}(\mathbf{x}_J\text{ - }\mathbf{x}_I) \\ &= \delta_{IJ} + \mathbf{H}^T(\mathbf{x}_J\text{ - }\mathbf{x}_I)\mathbf{Q}^{-1}(\mathbf{x}_J)[\mathbf{H}(\mathbf{0})\text{ - }\hat{\mathbf{F}}(\mathbf{x}_J)]\overline{\Phi}_{\overline{a}_I}(\mathbf{x}_J\text{ - }\mathbf{x}_I) = \delta_{IJ}\end{aligned} \tag{43}$$

Note that in Eq. (43) the property $\hat{a}_I < \min\{\|\mathbf{x}_I - \mathbf{x}_J\|, \forall J \neq I\}$ has been used. The RK interpolation function in Eq. (42) bares the following properties:

(i) $Supp(\Psi_I(\mathbf{x})) = \max(\overline{a}_I, \hat{a}_I)$

(ii) If $\overline{\Phi}_{a_I} \in C^{\overline{m}}$, $\hat{\Phi}_{\hat{a}_I} \in C^{\hat{m}}$, then $\Psi_I \in C^k$, $k = \min(\overline{m}, \hat{m})$

(iii) The singularity of $\mathbf{Q}(\mathbf{x})$ is only dependent on $\overline{a}_I$ and the order of basis function in $\mathbf{G}(\mathbf{x}\text{ - }\mathbf{x}_I)$, and is independent to $\hat{a}_I$.

(iv) For better accuracy in solving PDE's, the primitive functions are included only in the shape functions associated with the nodes on the essential boundary. Following Chen *et al.*[24], it can be shown that the coefficients of shape functions with primitive functions included are nodal values and essential boundary conditions can be imposed directly.

2.4 Stabilized Conformation Nodal Integration (SCNI)

2.4.1 Integration Constraints

The traditional approach to perform domain integration is Gauss integration. However, if Gauss quadrature is employed for integrating the weak form, an

additional background grid is required, and higher order quadrature rule is required to reduce the integration error. Another drawback of Gauss integration for the meshfree weak form is that it does not satisfy the integration constraints,[25] therefore the first order accuracy is not guaranteed even if the approximation of test and trial functions is linearly complete. Integration constraints are necessary conditions for linear exactness in the Galerkin approximation as identified by Chen *et al.*[25] There are two requirements for linear exactness in the 2nd order differential equations. The first condition, related to the approximation, requires the shape function to satisfy the linear consistency conditions given by

$$\begin{cases} \sum_{I=1}^{NP} \Psi_I(\mathbf{x}) = 1 \\ \sum_{I=1}^{NP} \Psi_I(\mathbf{x})\mathbf{x}_I = \mathbf{x} \end{cases}. \tag{44}$$

Note that $\mathbf{x}$ and $\mathbf{x}_I$ are vectors. These conditions are automatically satisfied in the reproducing kernel shape functions if complete linear basis functions are used. The second condition requires the integration of the gradient of the shape function to vanish if the shape function does not intersect with the boundary,[25] i.e.,

$$\sum_{L=1}^{NIT} \nabla\Psi_I(\mathbf{x}_L) w_L = \mathbf{0} \quad \text{if} \quad \text{supp}(\Psi_\mathbf{I}) \bigcap \partial\Omega = \varnothing, \tag{45}$$

where NIT is the number of integration points. If Gauss integration is employed, $\mathbf{x}_L$ are the spatial coordinates of the Gauss points and w_L are the weights of integration. If nodal integration is applied, $\mathbf{x}_L$ are the coordinates of the discrete integration points and w_L are the associated weights at the discrete points.

For shape function that intersects with the natural boundary, the integration of the gradient of the shape function should satisfy the divergence equation

$$\sum_{L=1}^{NIT} \nabla\Psi_I(\mathbf{x}_L) w_L = \sum_{K=1}^{NITB} \mathbf{n}\Psi_I(\mathbf{x}_K) s_K \quad \text{if} \quad \text{supp}(\Psi_I) \bigcap \Omega^h \neq \varnothing, \tag{46}$$

where $NITB$ is the number of integration points on the natural boundary that are covered by the support of node I, $\mathbf{n}$ is the outward normal of the natural boundary, and s_K are the weights of boundary integration.

2.4.2 Strain Smoothing

To satisfy the integration constraints as stated in Eqs. (45) and (46), a strain smoothing[25] is introduced

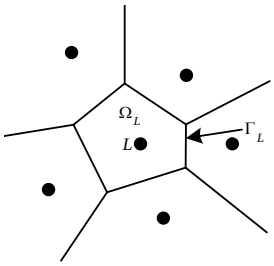


Figure 3. Nodal representative domain for SCNI

$$\begin{aligned}\bar{\varepsilon}_{ij}^{h}(\mathbf{x}_L) &= \frac{1}{2A_L}\int_{\Omega_L}\left(u_{i,j}^{h}+u_{j,i}^{h}\right)d\Omega = \frac{1}{2A_L}\int_{\Gamma_L}\left(u_i^h n_j + u_j^h n_i\right)d\Gamma \\ &= \frac{1}{2}[\bar{b}_{iI}(\mathbf{x}_L)d_{jI} + \bar{b}_{jI}(\mathbf{x}_L)d_{iI}]\end{aligned} \tag{47}$$

$$\bar{b}_{iI}(\mathbf{x}_L) = \frac{1}{A_L}\int_{\Gamma_L}\Psi_I(\mathbf{x})n_i d\Gamma\,. \tag{48}$$

Here Ω_L is the nodal representative domain for node L as shown in Fig. 3, Γ_L is the boundary of the representative domain, and A_L is the volume (for 3D) or area (for 2D) of the representative domain. A Voronoi diagram at particle L as shown in Fig. 3 can be employed to generate the nodal representative domain.

The smoothed strain approximation can be expressed as

$$\bar{\boldsymbol{\varepsilon}}^h(\mathbf{x}_L) = \sum_{I=1}^{NP}\bar{\mathbf{B}}_I(\mathbf{x}_L)\mathbf{d}_I \tag{49}$$

$$\bar{\mathbf{B}}_I(\mathbf{x}_L) = \begin{bmatrix}\bar{b}_{I1}(\mathbf{x}_L) & 0\\ 0 & \bar{b}_{I2}(\mathbf{x}_L)\\ \bar{b}_{I2}(\mathbf{x}_L) & \bar{b}_{I1}(\mathbf{x}_L)\end{bmatrix}, \tag{50}$$

where NP is the number of nodes whose support covers node $\mathbf{x}_L$. It has been shown that the smoothed gradient matrix $\bar{\mathbf{B}}$ satisfies the integration constraints.[25]

To introduce the smoothed strain into strain approximation, consider the following assumed strain variational equation:

$$\int_{\Omega_x}\delta\bar{\varepsilon}_{ij}C_{ijkl}\bar{\varepsilon}_{kl}d\Omega - \int_{\Omega}\delta u_i b_i d\Omega - \int_{\partial\Omega^h}\delta u_i h_i d\Gamma = 0\,. \tag{51}$$

By employing the smoothed strain approximation in Eq. (49) and the displacement approximation in Eq. (1), we have the discrete equation:

$$\mathbf{K}\mathbf{d} = \mathbf{f} \tag{52}$$

$$\mathbf{K}_{IJ} = \sum_{M=1}^{NP} \bar{\mathbf{B}}_I^T(\mathbf{x}_M)\mathbf{C}\bar{\mathbf{B}}_J(\mathbf{x}_M)A_M \,,\ \mathbf{f}_I = \sum_{M=1}^{NP} \Psi_I(\mathbf{x}_M)\mathbf{b}A_M + \sum_{L=1}^{Nh} \Psi_I(\mathbf{x}_L)\mathbf{h}s_L \quad (53)$$

2.5 Numerical Examples

2.5.1 Beam Subjected to a Shear Load

The problem statement and boundary conditions of the beam problem are given in Fig. 4 (a). The numerical solution obtained from SCNI is compared with the solutions obtained by Gauss integration with 5x5 quadrature rule and the direct nodal integration. Linear basis functions and a normalized support size of 2.01 are used in all three uniform discretizations. The comparison of displacement L_2 error norm is shown in Fig. 4 (b). The solution of the direct nodal integration presents lower accuracy than that obtained from Gauss integration. SCNI not

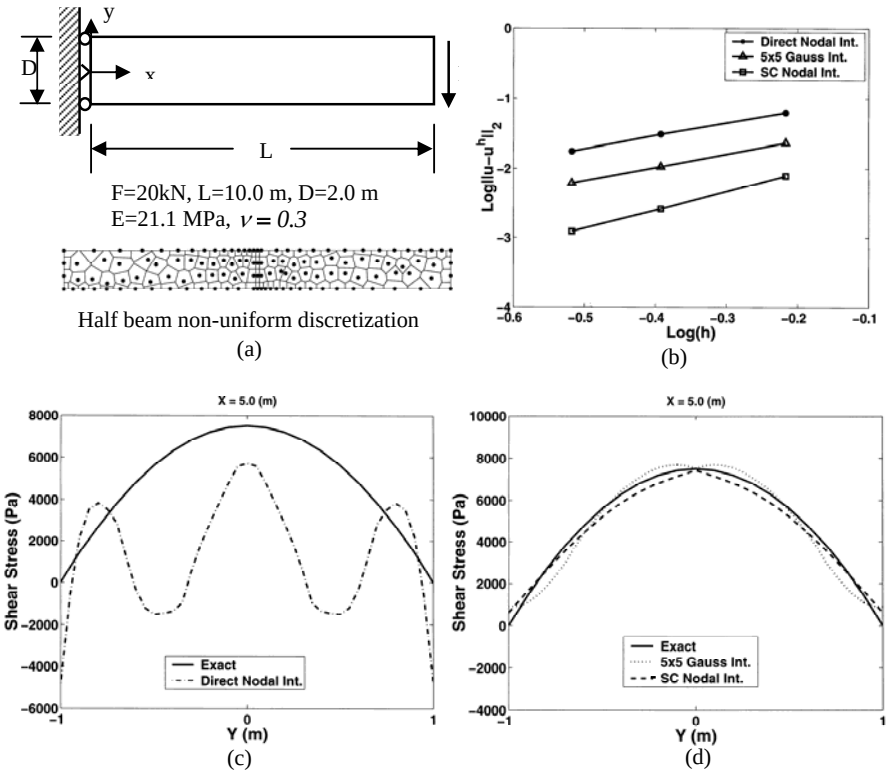


Figure 4. (a) Problem statement and discretization, (5) displacement L2 error norm, (c) shear stress distribution along x=0.5 L obtained by a direct nodal integration, (d) shear stress distribution along x=0.5 L obtained by the SCNI and 5x5 Gauss integration

Table 1 Tip displacement accuracy (%) using highly irregular discretization

Discrete Model	5x5 Gauss Int.	Direct Nodal Int.	SC Nodal Int.
124 nodes	94.99	192.82	99.25

only enhances accuracy of the direct nodal integration, the method performs better than the Gauss integration method. Shear stress distributions along regular nodes at $x = 0.5L$ in Figs. 4(c) and 4(d) clearly demonstrate the superior performance of SCNI. A very non uniform 124-node model as shown in Fig. 4 (a) is created to test the performance of three methods under highly non uniform discretization. The tip displacement solution of the direct nodal integration method deteriorates significantly in this case as shown in Table 1. On the contrary, SCNI still maintains a 99.25 % accuracy in the tip displacement; much better than direct nodal and Gauss integration methods.

3. Structural Shape Optimization

Structural design problems can be categorized based on the type of design variables. While the sizing design is related to parameters of the structure, the shape design is related to the structure's geometry. In the shape design problem, the structural domain or its boundary is defined as design variables. Since the domain itself is part of a design, the structural geometry appears implicitly as the design parameter. This fact makes the shape design problem more difficult than the conventional sizing design problem.

3.1 Shape Design Parameterization and Design Velocity

Shape design parameterization, which describes the boundary shape of a structure as a function of the design variables, is an essential step in the shape design process. Inappropriate parameterization can lead to unacceptable shapes. To parameterize the structural boundaries and to achieve optimum shape design, boundary shape can be described in three ways: (1) by using boundary nodal coordinates, (2) by using polynomials,[26-28] and (3) by using spline blending functions.[19,29-32] All these methods describe how the design variable changes the shape of the boundary.

In the meshfree method, the structural domain is discretized by a set of particles. When the boundary of the structure is changed according to the shape design variable, the location of meshfree particles is changed accordingly. The direction that each particle moves with respect to the shape design variable is called the design velocity. Let $\mathbf{x}$ be the location of a particle in the domain and the location at the perturbed design be given as

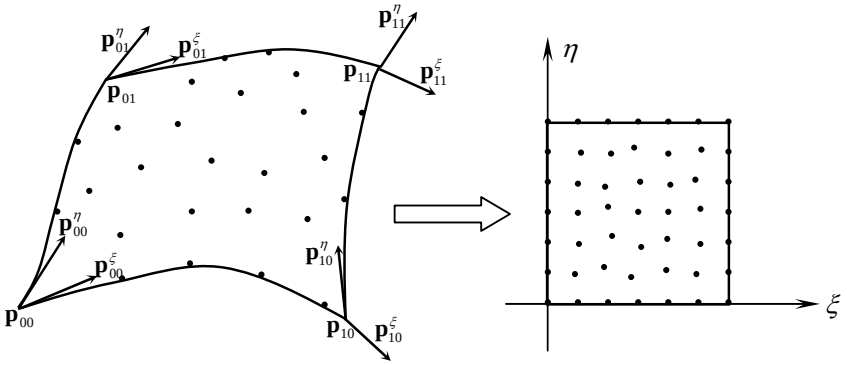


Figure 5. Parametric representation of a surface geometry. Corner points and their tangent vectors can be served as shape design variables. The parametric coordinates remain constant.

$$\mathbf{x}_{\tau} = \mathbf{x} + \tau \mathbf{V}(\mathbf{x}), \tag{54}$$

where $\mathbf{V}(\mathbf{x})$ is the design velocity and the parameter τ controls the magnitude of design change. The process is similar to the dynamic process by considering τ as time. Because of this analogy, the direction $\mathbf{V}(\mathbf{x})$ is called the design velocity.

In order to illustrate the shape design change, we consider a simple geometric representation as an example. In many geometric modelers, the location of particles is often represented using a parametric technique. For example, the location $\mathbf{x}$ in two–dimensional space can be represented using two parameters as

$$\mathbf{x}(\xi, \eta) = \mathbf{U}(\xi)^T \mathbf{M} \mathbf{G} \mathbf{M}^T \mathbf{W}(\eta), \tag{55}$$

where $\mathbf{U}(\xi) = [\xi^3 \quad \xi^2 \quad \xi \quad 1]^T$ and $\mathbf{W}(\eta) = [\eta^3 \quad \eta^2 \quad \eta \quad 1]^T$ are vectors in the parametric coordinates, and $\mathbf{M}$ is a constant matrix defined as

$$\mathbf{M} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{56}$$

and $\mathbf{G}$ is the geometric matrix defined as

$$\mathbf{G} = \begin{bmatrix} \mathbf{p}_{00} & \mathbf{p}_{01} & \mathbf{p}^{\eta}_{00} & \mathbf{p}^{\eta}_{01} \\ \mathbf{p}_{10} & \mathbf{p}_{11} & \mathbf{p}^{\eta}_{10} & \mathbf{p}^{\eta}_{11} \\ \mathbf{p}^{\xi}_{00} & \mathbf{p}^{\xi}_{01} & \mathbf{p}^{\xi\eta}_{00} & \mathbf{p}^{\xi\eta}_{01} \\ \mathbf{p}^{\xi}_{10} & \mathbf{p}^{\xi}_{11} & \mathbf{p}^{\xi\eta}_{10} & \mathbf{p}^{\xi\eta}_{11} \end{bmatrix}_{4\times4\times3} \tag{57}$$

where $\mathbf{p}_{ij}$ are coordinates of the corner points on the surface, $\mathbf{p}^{\xi}_{ij}$ and $\mathbf{p}^{\eta}_{ij}$ are the tangent vectors in ξ and η directions, respectively, and $\mathbf{p}^{\xi\eta}_{ij}$ is the twist vectors.

All components or combination of them can be served as shape design variables. Figure 5 shows the geometry and its transformation into the parametric coordinate.

The computation of $\mathbf{V}(\mathbf{x}) = \mathbf{V}(\xi,\eta)$ is directly related to the parametric representation of the neutral surface, as given in Eq. (55). For the purpose of explanation, let us consider one design variable b. Equation (55) is rewritten with design dependence as

$$\mathbf{x}(b;\xi,\eta) = \mathbf{U}(\xi)^T \mathbf{M}\mathbf{G}(b)\mathbf{M}^T\mathbf{W}(\eta)\,. \tag{58}$$

Since geometric matrix $\mathbf{G}(b)$ is a function of the design, the design velocity $\mathbf{V}(\xi,\eta)$ can be obtained by perturbing b to $b+\tau\delta b$, and then differentiating with respect to τ as

$$\mathbf{V}(\xi,\eta) = \left.\frac{d\mathbf{x}(b+\tau\delta b;\xi,\eta)}{d\tau}\right|_{\tau=0} = \mathbf{U}(\xi)^T\mathbf{M}(\frac{\partial\mathbf{G}}{\partial b}\delta b)\mathbf{M}^T\mathbf{W}(\eta)\,. \tag{59}$$

For example, when the *x*–component of $\mathbf{p}_{00}$ is chosen as the design variable, matrix $\partial\mathbf{G}/\partial b$ has all zero components except for the component at (1,1) that has a value of [1, 0, 0]. The design velocity field must be obtained per each shape design variable.

An advantage of the design velocity computation in Eq. (59) is that it is unnecessary to store design velocity for all particles. It is sufficient to simply store matrix $\partial\mathbf{G}/\partial b$ for each design variable. Note that $\mathbf{V}(\xi,\eta)$ remains constant during the optimization process.

3.2 Shape Sensitivity Analysis

Design sensitivity analysis computes the rate of performance measure change with respect to design variable changes.[33] With the structural analysis, the design sensitivity analysis generates a critical information, gradient, for optimization. In this text, performance measures are presumed to be differentiable with respect to design, at least in the neighborhood of the current design point. For complex engineering applications, however, it is not simple to prove the differentiability.[34]

In general, a structural performance measure depends on the design parameters. For example, a change in the cross-sectional area of a beam would affect the structural weight. This type of dependence is simple if the expression of weight in terms of the design variables is known. This type of function is *explicitly dependent* on the design. Consequently, only algebraic calculation is involved to obtain the design sensitivity of an explicitly dependent performance measure.

However, in many cases, performance measures implicitly depend on the design. For example, there is no explicit way to express the stress of a structure

explicitly in terms of the design variable b. Consider the general performance measure $\psi = \psi(\mathbf{u}(b), b)$ that depends on the design explicitly and implicitly. The sensitivity of ψ can be expressed as

$$\frac{d\psi(\mathbf{u}(b),b)}{db} = \left.\frac{\partial \psi}{\partial b}\right|_{\mathbf{u}=\text{const}} + \left.\frac{\partial \psi}{\partial \mathbf{u}}\right|_{b=\text{const}}^{T} \frac{d\mathbf{u}}{db}. \tag{60}$$

From the expression of $\psi(\mathbf{u}(b), b)$, the explicitly dependent term, $d\psi / db$, and the derivative, $d\psi / d\mathbf{u}$, can easily be obtained. The only unknown term in Eq. (60) is $d\mathbf{u} / db$, which is the sensitivity of the state variable with respect to the design variable. The key procedure of design sensitivity analysis is to calculate the sensitivity of the state variable by differentiating the structural equation. For a given shape design velocity field $\mathbf{V}(\mathbf{x})$, the shape sensitivity formulation expresses the sensitivity of state variable in terms of the design velocity. In this text, only linear problem is considered. The nonlinear sensitivity analysis is presented in Chapter 8.

Four approaches are used to obtain design sensitivity: the finite difference, discrete, continuum, and computational derivatives. In the finite difference approach, design sensitivity is obtained by either the *forward* or *central finite difference method*. In the discrete method, design sensitivity is obtained by taking design derivatives of the discrete governing equation. In the continuum approach, the design derivative of the variational equation is taken before discretization. Finally, computational or automatic differentiation refers to a differentiation of the computer code itself. The continuum approach is employed in this text because this formulation is independent of discretization methods. The particular discretization using the meshfree method will be discussed in the next section.

First, the sensitivity of the state variable is defined using the material derivative concept in continuum mechanics, as

$$\begin{aligned}
\dot{\mathbf{u}}(\mathbf{x}) &= \lim_{\tau\to 0}\left[\frac{\mathbf{u}_\tau(\mathbf{x}+\tau\mathbf{V}(\mathbf{x})) - \mathbf{u}(\mathbf{x})}{\tau}\right] \\
&= \lim_{\tau\to 0}\left[\frac{\mathbf{u}_\tau(\mathbf{x}) - \mathbf{u}(\mathbf{x})}{\tau}\right] + \lim_{\tau\to 0}\left[\frac{\mathbf{u}_\tau(\mathbf{x}+\tau\mathbf{V}(\mathbf{x})) - \mathbf{u}_\tau(\mathbf{x})}{\tau}\right], \\
&= \mathbf{u}'(\mathbf{x}) + \nabla\mathbf{u}\cdot\mathbf{V}(\mathbf{x}),
\end{aligned} \tag{61}$$

where $\mathbf{u}'$ is the partial derivative. The above material derivative can be applied to general functions. The order of differentiation can be changed between the partial derivative and the spatial derivative, such that $(\nabla\mathbf{u})' = \nabla(\mathbf{u}')$. However, it is not true for the material derivative in Eq. (61). In such a case,

$$(\nabla\mathbf{u})^{\cdot} = \nabla\dot{\mathbf{u}} - \nabla\mathbf{u}\cdot\nabla\mathbf{V}. \tag{62}$$

Since the structural equation is expressed in terms of functionals, the following two formulas are useful for deriving the sensitivity equation:

$$\psi_1' = \left[\int_\Omega f(\mathbf{x})\, d\Omega \right]' = \int_\Omega [\dot{f}(\mathbf{x}) + f(\mathbf{x}) div\mathbf{V}]\, d\Omega \tag{63}$$

$$\psi_2' = \left[\int_{\partial\Omega} g(\mathbf{x})\, d\Gamma \right]' = \int_{\partial\Omega} [\dot{g}(\mathbf{x}) + \kappa g(\mathbf{x}) V_n]\, d\Gamma\,. \tag{64}$$

In Eq. (64), κ is the curvature of the boundary and V_n is the normal component of the design velocity on the boundary.

The variational equation in Eq. (30) is used for deriving the sensitivity equation. For the illustration, Eq. (30) is rewritten in the following form:

$$\begin{aligned} a_\Omega(\mathbf{u}^h, \delta\mathbf{u}^h) &\coloneqq \int_\Omega \delta(\nabla^S u^h)_{ij} C_{ijkl} (\nabla^S u^h)_{kl}\, d\Omega \\ &= \int_\Omega \delta u_i^h b_i d\Omega + \int_{\partial\Omega^h} \delta u_i^h h_i d\Gamma \coloneqq \ell_\Omega(\delta\mathbf{u}^h) \end{aligned}\,. \tag{65}$$

The notation is selected such that $a_\Omega(\mathbf{u}, \delta\mathbf{u})$ is bilinear with respect to its two arguments, while $\ell_\Omega(\delta\mathbf{u})$ is linear. The above variational equation must satisfy for all kinematically admissible fields $u_i^h \in H_g^1$ and $\delta u_i^h \in H_0^1$.

Using the formulas in Eqs. (63) and (64), the above variational equation is differentiated to obtain the sensitivity equation:

$$a_\Omega(\dot{\mathbf{u}}^h, \delta\mathbf{u}^h) = \ell'_{\mathbf{V}}(\delta\mathbf{u}^h) - a'_{\mathbf{V}}(\mathbf{u}^h, \delta\mathbf{u}^h)\,, \tag{66}$$

for all $\delta u_i^h \in H_0^1$. In Eq. (66), the left-hand side is identical with that of Eq. (65) if $\mathbf{u}^h$ is substituted with its sensitivity $\dot{\mathbf{u}}^h$, and two terms on the right-hand side are defined as

$$\begin{aligned} \ell'_{\mathbf{V}}(\delta\mathbf{u}^h) = &\int_\Omega [\delta u_i^h (\nabla b)_{ij} V_j + \delta u_i^h b_i div\mathbf{V}]\, d\Omega \\ &+ \int_{\partial\Omega^h} [\delta u_i^h (\nabla h)_{ij} V_j + \kappa \delta u_i^h h_i V_n]\, d\Gamma \end{aligned} \tag{67}$$

and

$$\begin{aligned} a'_{\mathbf{V}}(\mathbf{u}^h, \delta\mathbf{u}^h) = \int_\Omega [&\varepsilon_{ij}^{\mathbf{V}}(\delta\mathbf{u}^h) C_{ijkl} (\nabla^S u^h)_{kl} + \delta(\nabla^S u^h)_{ij} C_{ijkl} \varepsilon_{kl}^{\mathbf{V}}(\mathbf{u}^h) \\ &+ \delta(\nabla^S u^h)_{ij} C_{ijkl} (\nabla^S u^h)_{kl}\, div\mathbf{V}]\, d\Omega \end{aligned}\,, \tag{68}$$

where $div\mathbf{V} = \partial V_i / \partial x_i$, and

$$\varepsilon_{ij}^{\mathbf{V}}(\mathbf{u}^h) = -\frac{1}{2}\left(\frac{\partial u_i^h}{\partial x_k}\frac{\partial V_k}{\partial x_j} + \frac{\partial u_j^h}{\partial x_k}\frac{\partial V_k}{\partial x_i}\right). \tag{69}$$

The detailed derivations can be found in Choi and Seong.[35]

It is well known that the adjoint variable method is more efficient than solving the design sensitivity equation (66) when the number of design variables is greater than the number of performance functions. However, since this paper aims to address the dependence between shape design variables and the meshfree approximation function, the discussion will be limited to the direct differentiation method as in Eq. (66).

The shape sensitivity equation (66) is independent of discretization method. Either finite element[36] or meshfree method[37] can be used for numerically calculating the sensitivity of the state variable. In the following section, the implementation using the meshfree method is discussed.

3.3 Meshfree Discretization of Sensitivity Equation

3.3.1 Material Derivative of Meshfree Shape Function

Since the main unknown variable of the meshfree method is generalized displacement d_{iI}, the design sensitivity equation (66) in the continuum form, which is written in terms of $\dot{u}_i^h$, has to be discretized using $\dot{d}_{iI}$. Since u_i^h is approximated using the meshfree shape function in Eq. (22), $\dot{u}_i^h$ can be approximated by differentiating Eq. (22), as

$$\dot{u}_i^h(\mathbf{x}) = \sum_{I=1}^{NP}(\Psi_I(\mathbf{x})\dot{d}_{iI} + \dot{\Psi}_I(\mathbf{x})d_{iI}). \tag{70}$$

This decomposition is quite different from the finite element method in which the shape function is independent of the design. The first term constitutes the main unknown $\dot{d}_{iI}$ of the sensitivity equation, while the second represents the dependence of the shape function on design, which is explicit in $\mathbf{V}(\mathbf{x})$.

From the observation that u_i^h and $\dot{u}_i^h$ belong to the same space,† $\dot{u}_i^h$ can be approximated directly using the meshfree shape function[38] as

$$\dot{u}_i^h(\mathbf{x}) = \sum_{I=1}^{NP}\Psi_I(\mathbf{x})\tilde{d}_{iI}. \tag{71}$$

By comparing Eq. (70) with Eq. (71), the latter seems to provide simpler approximation than the former. However, the former will yield numerical results that are more consistent than the latter. In addition, since $\tilde{d}_{iI}$ is not the material

† This can be observed by comparing Eq. (65) with Eq. (66).

derivative of d_{iI}, the penalty function must be used for imposing the essential boundary conditions. In this text, the approximation in Eq. (70) will be used.

A numerical method to compute $\dot{\Psi}_I(\mathbf{x})$ will now be introduced. From the relation $\mathbf{x}_\tau = \mathbf{x} + \tau\mathbf{V}(\mathbf{x})$, the derivative of the material point $\mathbf{x}$ is nothing but the design velocity $\mathbf{V}(\mathbf{x})$. Consider the material derivative of the kernel function in Eq. (3) for a one–dimensional problem,

$$\dot{\Phi}_a(x - x_I) = \frac{4(V_I - V)}{a}\begin{cases} 2z - 3z^2, & z \le \frac{1}{2} \\ (1-z)^2, & \frac{1}{2} < z \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{72}$$

where V_I is the design velocity at x_I, and V is the design velocity at x. For a multi–dimensional problem, the product rule in Eq. (5) can be used.

To compute $\dot{\Psi}_I(\mathbf{x})$, the material derivative of the reproducing condition in Eq. (11) has to be taken, to obtain

$$\dot{\mathbf{b}}(\mathbf{x}) = -\mathbf{M}(\mathbf{x})^{-1}\dot{\mathbf{M}}(\mathbf{x})\mathbf{b}(\mathbf{x})\,, \tag{73}$$

where,

$$\dot{\mathbf{M}}(\mathbf{x}) = \sum_{I=1}^{NP}[\dot{\mathbf{H}}\mathbf{H}^T\Phi_a + \mathbf{H}\dot{\mathbf{H}}^T\Phi_a + \mathbf{H}\mathbf{H}^T\dot{\Phi}_a] \tag{74}$$

$$\dot{\mathbf{H}}(\mathbf{x} - \mathbf{x}_I) = \begin{bmatrix} 0 & V_1 - V_{1I} & V_2 - V_{2I} & \cdots & n(x_2 - x_{2I})^{n-1}(V_2 - V_{2I}) \end{bmatrix}^T. \tag{75}$$

Thus, from the definition of the meshfree shape function in Eq. (15), we have

$$\dot{\Psi}_I(\mathbf{x}) = \dot{\mathbf{b}}^T\mathbf{H}\Phi_a + \mathbf{b}^T\dot{\mathbf{H}}\Phi_a + \mathbf{b}^T\mathbf{H}\dot{\Phi}_a\,. \tag{76}$$

For given design velocity $\mathbf{V}(\mathbf{x})$, Eq. (76) can be explicitly calculated even before any sensitivity analysis. The material derivative of $d\Psi_I / d\mathbf{x}$ can also be calculated using a similar procedure.

3.3.2 Discrete Form of Sensitivity Equation

In developing sensitivity formulation, it is necessary to take the material derivative of strain or, equivalently, the gradient of displacement, $u^h_{i,j} = \partial u^h_i / \partial x_j$. Choi and Kim[33] uses the concept of a partial derivative that is commutable to a spatial gradient. By using Eqs. (62) and (70), a meshfree approximation of the material derivative of $u^h_{i,j}$ can be expressed as

$$\left(u^h_{i,j}\right)^{\bullet} = \sum_{I=1}^{NP}(\Psi_{I,j}\dot{d}_{iI} + \dot{\Psi}_{I,j}d_{iI} - \Psi_{I,k}d_{iI}V_{k,j})\,. \tag{77}$$

Note that the last two terms are explicitly dependent on the design velocity. The only unknown term is $\dot{d}_{iI}$ which will be computed from the sensitivity equation. To simplify the approximation of Eq. (77), the following relation can be used:

$$\left(\Psi_{I,j}\right)^{\bullet} = \dot{\Psi}_{I,j} - \Psi_{I,k}V_{k,j}\,. \tag{78}$$

Thus, the last two terms of Eq. (77) can be combined to represent an explicitly dependent term on $\mathbf{V}(\mathbf{x})$ through $(\Psi_{I,i})^{\bullet}$. Using Eq. (78), Eq. (77) is simplified to

$$\left(u^h_{i,j}\right)^{\bullet} = \sum_{I=1}^{NP}\Psi_{I,j}\dot{d}_{iI} + \sum_{I=1}^{NP}\left(\Psi_{I,j}\right)^{\bullet} d_{iI}\,. \tag{79}$$

Note that the two summations of Eq. (79) have a similar format. The first term on the right–hand side has to be solved using a design sensitivity equation, and the second term can be computed explicitly using the design velocity.

In sensitivity analysis, it is often assumed that the space H^1_0 of kinematically admissible displacements is independent of shape design, i.e., $\delta\dot{u}^h_i = 0$. Even if the assumption of $\delta\dot{u}^h_i = 0$ is not used, since $\delta\dot{u}^h_i \in H^1_0$, the following relation is satisfied

$$a_{\Omega}(\mathbf{u}^h, \delta\dot{\mathbf{u}}^h) = \ell_{\Omega}(\delta\dot{\mathbf{u}}^h)\,. \tag{80}$$

Because of Eq. (80), the contribution of $\delta\dot{u}^h_i$ will be ignored in the derivation of sensitivity equation. In addition, from the relation in Eq. (77),

$$\left(\delta u^h_{i,j}\right)^{\bullet} = -\delta u^h_{i,k}V_{k,j} = -\sum_{I=1}^{NP}\Psi_{I,k}\delta d_{iI}V_{k,j}\,. \tag{81}$$

The approximation of the sensitivity equation (66) follows the same method as meshfree analysis. For a given meshfree shape function, using its material derivatives from Eq. (76), as well as using the relation in Eq. (79), the following approximation can be obtained:

$$\boldsymbol{\varepsilon}^{\mathbf{V}}(\mathbf{u}^h) = \sum_{I=1}^{NP}\dot{\mathbf{B}}_I\mathbf{d}_I\,, \tag{82}$$

where $\dot{\mathbf{B}}_I$ is the material derivative of $\mathbf{B}_I$, defined by

$$\dot{\mathbf{B}}_I = \begin{bmatrix} \left(\Psi_{I,1}\right)^{\bullet} & \left(\Psi_{I,2}\right)^{\bullet} & 0 \\ 0 & \left(\Psi_{I,1}\right)^{\bullet} & \left(\Psi_{I,2}\right)^{\bullet} \end{bmatrix}^T \tag{83}$$

In contrast, the approximation of $\boldsymbol{\varepsilon}^V(\delta\mathbf{u}^h)$ has a different expression because of Eq. (81),

$$\boldsymbol{\varepsilon}^{\mathbf{V}}(\delta\mathbf{u}^h) = \sum_{I=1}^{NP}\mathbf{B}^{\mathbf{V}}_I\delta\mathbf{d}_I \tag{84}$$

$$\mathbf{B}_I^{\mathbf{V}} = -\begin{bmatrix} \Psi_{I,k}V_{k,1} & \Psi_{I,k}V_{k,2} & 0 \\ 0 & \Psi_{I,k}V_{k,1} & \Psi_{I,k}V_{k,2} \end{bmatrix}^T . \tag{85}$$

Now, the right-hand sides of sensitivity equation, Eqs. (67) and (68), can be approximated by

$$\begin{aligned} \ell_V'(\delta\mathbf{u}^h) \approx & \int_\Omega \sum_{I=1}^{NP} \delta d_{iI}\Psi_I[(\nabla b)_{ij}V_j + b_i div\mathbf{V})]d\Omega \\ & + \int_{\partial\Omega^h} \sum_{I=1}^{NP} \delta d_{iI}\Psi_I[(\nabla h)_{ij}V_j + \kappa h_i V_n)]d\Omega \equiv \delta\mathbf{d}^T\mathbf{F}^\ell \end{aligned} \tag{86}$$

$$a_V'(\mathbf{u}^h, \delta\mathbf{u}^h) \approx \int_\Omega \sum_{I=1}^{NP} \delta\mathbf{d}_I^T[\mathbf{B}_I^{V^T}\boldsymbol{\sigma} + \mathbf{B}_I^T\mathbf{C}\boldsymbol{\varepsilon}^{\mathbf{V}}(\mathbf{u}^h) + \mathbf{B}_I^T\boldsymbol{\sigma}\, div\mathbf{V}]d\Omega \equiv \delta\mathbf{d}^T\mathbf{F}^a . \tag{87}$$

Thus, the discrete form of the sensitivity equation becomes

$$\delta\mathbf{d}^T\mathbf{K}\dot{\mathbf{d}} = \delta\mathbf{d}^T(\mathbf{F}^\ell - \mathbf{F}^a) , \tag{88}$$

for all $\delta\mathbf{d}$ whose counterparts $\delta\mathbf{u}$ belong to the space H_0^1 of kinematically admissible displacements.

3.3.3 Imposing Essential Boundary Conditions

The discrete sensitivity equation (88) cannot be solved directly because it is not trivial to construct the kinematically admissible $\delta\mathbf{d}$ from Eq. (88). As discussed in Section 2, the Lagrange multiplier method in Eq. (20) can be used for the purpose of sensitivity analysis. In such a case, the sensitivity of the Lagrange multiplier also needs to be calculated. In addition, the coefficient matrix becomes positive semi-definite, which requires a special treatment in solving the matrix equation. When the modified reproducing kernel approximation is used, Eq. (88) can directly be used because the modified meshfree shape functions for the boundary particles satisfy the interpolation property. Thus, the transformation method will be discussed in the following.

By following the same response analysis procedure to construct kinematically admissible displacements, the following linear matrix equation is solved:

$$\mathbf{K}^*\dot{\mathbf{d}} = \boldsymbol{\Lambda}^{*-T}(\mathbf{F}^\ell - \mathbf{F}^a) , \tag{89}$$

where $\mathbf{K}^*$ represents the same stiffness matrix with meshfree analysis as in Eq. (32), which is already factorized. Thus, it is very efficient to solve (89) with different right-hand sides.

Consideration of the essential boundary conditions is somewhat different from that of the analysis undertaken in Eq. (33), since the transformation matrix

$\mathbf{\Lambda}^*$, which is composed of a meshfree shape function, depends on the shape design. Let the prescribed displacement $\mathbf{g}$ at $\mathbf{x} \in \partial\Omega^g$ be independent of design, which is true in most cases. Then, from $\hat{\mathbf{d}}^B = \mathbf{g}$ and Eq. (26), we have

$$\mathbf{\Lambda}^B \dot{\mathbf{d}} = -\dot{\mathbf{\Lambda}}^B \mathbf{d}\,, \tag{90}$$

where $\mathbf{\Lambda}^B = [\mathbf{\Lambda}^{BB} \quad \mathbf{\Lambda}^{BI}]$ and $\dot{\mathbf{\Lambda}}^B_{IJ} = \dot{\Psi}_I(\mathbf{x}_J)$ is obtained from Eq. (70) with $\mathbf{x}_J \in \partial\Omega^g$. Equation (90) is substituted into Eq. (89) for those rows that correspond to the particles on the essential boundary, and by the use of Eq. (33), we have

$$\begin{bmatrix} \mathbf{\Lambda}^{BB} & \mathbf{\Lambda}^{BI} \\ \mathbf{K}^{*IB} & \mathbf{K}^{*II} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{d}}^B \\ \dot{\mathbf{d}}^I \end{bmatrix} = \begin{bmatrix} -\dot{\mathbf{\Lambda}}^B \mathbf{d} \\ \left(\mathbf{\Lambda}^{*-T}(\mathbf{F}^\ell - \mathbf{F}^a)\right)^I \end{bmatrix}. \tag{91}$$

Equation (91) is solved for each design parameter with the same coefficient matrix with the meshfree analysis. After solving $\dot{\mathbf{d}}$, the material derivative of physical displacement can be calculated from the relation in Eq. (70).

4. Numerical Examples

4.1 Torque-Arm Model

The shape of the torque-arm model in Fig. 6 is optimized according to the eight shape design variables that control the boundary curves. In each design variable, the design velocity is calculated using Eq. (59). The torque-arm is modeled using 239 meshfree particles. Figure 6(a) shows meshfree particles and analysis results. The sensitivity computation requires only 10% of meshfree analysis computation per design variable due to the use of the same tangent operator as shown in Eqs. (33) and (91).

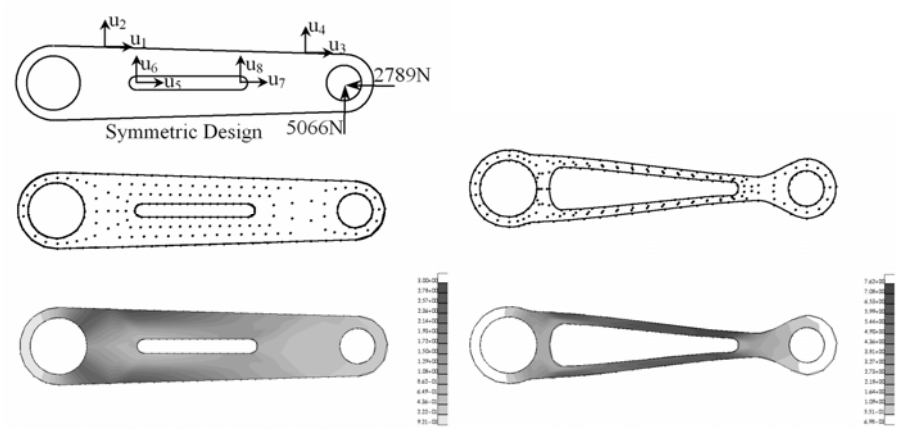
The design optimization problem is formulated to minimize the structural mass, with the effective stress constraint, as

$$\begin{aligned} &\text{minimize} \quad \text{mass} \\ &\text{subject to} \quad \sigma_{MAX} \leq 800 \quad \text{MPa} \end{aligned}. \tag{92}$$

The sequential quadratic programming method is used in a commercially available optimization program.[39] Figure 6(b) shows the meshfree analysis results at optimum design where the stress constraints along the upper side of torque arm is active. No re-modeling is used during the design optimization procedure. Through optimization, the structural mass is reduced by 48%. A total of 41 meshfree analyses and 20 sensitivity analyses are carried out during 20 optimization iterations. When finite element analysis is used with a re-meshing

process,[40] the optimization process converged at 45 iterations with eight re-meshing processes. Thus, this approach reduces the cost of design optimization more than 50%, leaves along the cost related to the re-meshing process.

Since the initial particle distribution is used throughout the optimization process, a very non uniform particle distribution is resulted in the optimum design. The analysis result from evenly distributed particles at the optimum design confirms that the solution accuracy is insensitive to the particle distribution.



(a) (b)

Figure 6. Design parameterization and meshfree analysis results of a torque-arm: (a) Initial design and (b) optimum design.

4.2 Road-Arm Model

The advantage of meshfree analysis for structural optimization is more significant in the case of three-dimensional problem. Figure 7 shows a road arm model that is discretized with 1,455 meshfree particles. Eight shape design variables are defined to optimize the shape of the boundary. Since the geometries in the corner are so complicated, it is challenging to construct a regular-shaped finite element mesh. In addition to the complicated initial geometry, the structural shape further changes during design optimization process, which will cause a mesh distortion problem if a finite element method is used.

As is illustrated in Figure 7, the stress concentration appears in the left corner of the road arm. If the highest stress level in the left corner is considered as a reference value, then the dimension of the right corner cross-section can be reduced, because this region has a large amount of safety margin.

The design optimization is carried out to minimize the structural weight of the road arm, while maintaining the maximum stress level. Design optimization problem converges after eight iterations. Figure 7 also compares the meshfree analysis result at the initial and optimum designs. The structural weight at the optimum design is reduced by 23% compared to the initial weight. Since the stress concentration appears at the left corner in the initial design, the optimization algorithm intends to reduce the cross-sectional area of the right corner so that both parts may have the same level of stress values. Because of the significant geometry changes at the right corner, the mesh distortion problem may occur if the finite element-based analysis method is employed.

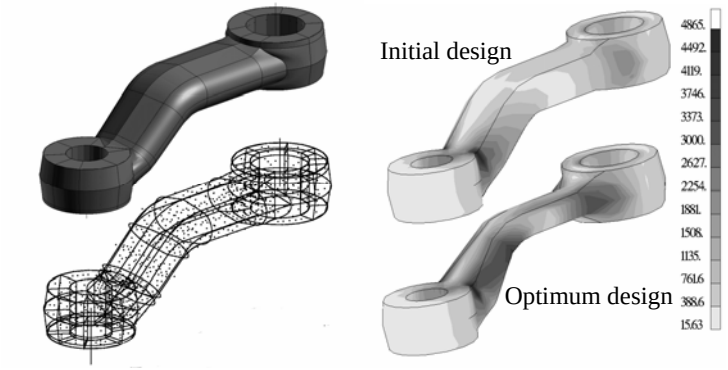


Figure 7. Meshfree discretization and analysis results of a road arm.

5. Summary and Conclusions

Design sensitivity analysis (DSA) and optimization based on meshfree method have been proposed. Unlike finite element and boundary element methods, in meshfree approach the shape function of the meshfree approximation depends on shape design parameterization, and this effect has been discussed in detail. DSA based on stabilized conforming nodal integration completely removes background mesh, and the integration of the shape DSA and optimization capability has been effectively carried out. It has also been shown that shape design optimization of structures undergoing large shape changes can be effectively carried out using meshfree methods without re-meshing. Fast convergence of the design optimization algorithm has been accomplished using the accurate sensitivity information.

References

1 R. A. Gingold and J. J. Monaghan, *Monthly Notices Royal Astro. Soc.* **181,** 375-389 (1977).
2 B. Nayroles, G. Touzot, and P. Villon, *Comp. Mech.* **10** (1992).
3 T. Belytschko, Y. Y. Lu, and L. Gu, *Int. J. Numer. Methods Eng.* **37,** 229-256 (1994).
4 W. K. Liu, S. Jun, and Y. F. Zhang, *Int. J. Numer. Methods Fluids* **20,** 1081-1106 (1995).
5 J. S. Chen, C. Pan, C. T. Wu, and W. K. Liu, *Comp. Methods Appl. Mech. Eng.* **139,** 195-227 (1996).
6 C. A. M. Duarte and J. T. Oden, *Comp. Methods Appl. Mech. Eng.* **139,** 237-262 (1996).
7 J. M. Melenk and I. Babuska, *Comp. Methods Appl. Mech. Eng.* **139,** 289-314 (1996).
8 N. Sukumar, B. Moran, and T. Belytschko, *Int. J. Numer. Methods Eng.* **43,** 839-887 (1998).
9 S. N. Atluri and T. Zhu, *Comp. Mech.* **25,** 169-179 (2000).
10 S. De and K. J. Bathe, *Comp. Mech.* **25,** 329-345 (2000).
11 T. Belytschko, Y. Y. Lu, and L. Gu, *Eng. Fracture Mech.* **51,** 295-315 (1995).
12 C. A. M. Duarte and J. T. Oden, *Numer. Methods Partial Diff. Equa.* **12,** 673-705 (1996).
13 I. Babuska and J. M. Melenk, *Int. J. Numer. Methods Eng.* **40,** 727-758 (1997).
14 Y. You, J. S. Chen, and H. Lu, *Comp. Mech.* **31,** 316-326 (2003).
15 J. S. Chen, C. Pan, C. M. O. Roque, and H. P. Wang, *Comp. Mech.* **22,** 289-307 (1998).
16 I. Grindeanu, K. K. Choi, and J. S. Chen, *AIAA J.* **37,** 990-1016 (1999).
17 N. H. Kim, K. K. Choi, and J. S. Chen, *AIAA J.* **38,** 1742-1753 (2000).
18 M. E. Botkin, *AIAA J.* **20,** 268-273 (1982).
19 T. M. Yao and K. K. Choi, *ASME J. Struct. Eng.* **115,** 2401-2405 (1989).
20 P. Lancaster and K. Salkauskas, *Math. Comp.* **37,** 141-158 (1981).
21 R. L. Hardy, *J. Geophysics Res.* **176,** 1905-1915 (1971).
22 J. Dolbow and T. Belytschko, *Comp. Mech.* **23,** 219-230 (1999).
23 J. S. Chen and H. P. Wang, *Comp. Methods Appl. Mech. Eng.* **187,** 441-468 (2000).
24 J. S. Chen, W. Han, Y. You, and X. Meng, *Int. J. Numer. Methods Eng.* **56,** 935-960 (2003).
25 J. S. Chen, C. T. Wu, S. Yoon, and Y. You, *Int. J. Numer. Methods Eng.* **50,** 435-466 (2001).
26 S. S. Bhavikatti and C. V. Ramakrishnan, *Comp. Struct.* **11,** 397-401 (1980).
27 E. S. Kristensen and N. F. Madsen, *Int. J. Numer. Methods Eng.* **10,** 1007-1019 (1976).
28 P. Pedersen and C. L. Laursen, *J. Struct. Mech.* **10,** 375-391 (1982-83).
29 R. J. Yang and K. K. Choi, *J. Struct. Mech.* **13,** 223-239 (1985).
30 M. L. Luchi, A. Poggialini, and F. Persiani, *Comp. Struct.* **11,** 629-637 (1980).
31 M. Weck and P. Steinke, *J. Struct. Mech.* **11,** 433-449 (1983-4).
32 T. M. Yao and K. K. Choi, *Int. J. Numer. Methods Eng.* **28,** 369-384 (1989).
33 K. K. Choi and N. H. Kim, *Structural Sensitivity Analysis and Optimization 1: Linear Systems* (Springer, New York, 2004).
34 E. J. Haug, K. K. Choi, and V. Komkov, *Design Sensitivity Analysis of Structural Systems* (Academic Press, London, 1986).
35 K. K. Choi and H. G. Seong, *Comp. Methods Appl. Mech. Eng.* **57,** 1-15 (1986).
36 K. H. Chang, K. K. Choi, C. S. Tsai, B. S. Choi, and X. M. Yu, *Comp. Sys. Eng.* **6,** 151-175 (1995).
37 N. H. Kim, K. K. Choi, and M. Botkin, *Struct. Multidiscipl. Optim.* **24,** 418-429 (2003).
38 F. Bobaru and S. Mukherjee, *Comp. Methods Appl. Mech. Eng.* **190,** 4319-4337 (2001).
39 G. N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design with Applications* (Vanderplaats Research & Development Inc., Colorado Springs, CO, 1999).
40 J. A. Bennett and M. E. Botkin, *AIAA J.* **23,** 458-464 (1985).