Implementing the Node Based Smoothed Finite Element Method as User Element in Abaqus for Linear and Nonlinear Elasticity

S. Kshrisagar[1], A. Francis[1], J. J. Yee[2], S. Natarajan[1] and C. K. Lee[3, *]

Abstract: In this paper, the node based smoothed-strain Abaqus user element (UEL) in the framework of finite element method is introduced. The basic idea behind of the node based smoothed finite element (NSFEM) is that finite element cells are divided into subcells and subcells construct the smoothing domain associated with each node of a finite element cell [Liu, Dai and Nguyen-Thoi (2007)]. Therefore, the numerical integration is globally performed over smoothing domains. It is demonstrated that the proposed UEL retains all the advantages of the NSFEM, i.e., upper bound solution, overly soft stiffness and free from locking in compressible and nearly-incompressible media. In this work, the constant strain triangular (CST) elements are used to construct node based smoothing domains, since any complex two dimensional domains can be discretized using CST elements. This additional challenge is successfully addressed in this paper. The efficacy and robustness of the proposed work is obtained by several benchmark problems in both linear and nonlinear elasticity. The developed UEL and the associated files can be downloaded from https://github.com/nsundar/NSFEM.

Keywords: Smoothed finite element method (SFEM), node based SFEM (NSFEM), linear and nonlinear elasticity, Abaqus UEL (user elements), compressible and nearly-incompressible materials.

1 Introduction

The classical FEM is a well established numerical method for simulation of real world problems related to science and engineering. The commercial finite element software like Abaqus is widely used in such applications. The Finite Element Analysis (FEA) involves three steps viz. pre-processing, analysis and post-processing. The pre-processing stage focuses on discretization of problem domain into non-overlapping sub-domains known as finite elements. This process of discretization is known as meshing. In classical FEM, shapes of the elements are restricted to triangles and quadrilaterals in two dimensions and tetrahedra and hexahedra in three dimensions [Altair Hypermesh Documentation (2014)].

[1] Integrated Modelling and Simulation Lab, Department of Mechanical Engineering, Indian Institute of Technology Madras, Chennai, 600036, India.

[2] The Department of Architectural Engineering, Dong-A University, S04-0302, Engineering 2, 37, Nakdong-daero 550beon-gil, Saha-gu, Busan, 49315, Korea.

[3] National Research Center for Disaster-free & Safe Ocean City, Dong-A University, S04-0311-3, Engineering 2, 37, Nakdong-daero 550beon-gil, Saha-gu, Busan, 49315, Korea.

[*] Corresponding Author: C. K. Lee. Email: changkyelee@dau.ac.kr.

Due to these restrictions on element topology, the accuracy of the classical FEM is highly influenced by mesh distortion [Lee and Bathe (1993)]. In this regard, the triangular elements are considered to be robust for meshing but not in terms of accuracy like quadrilateral elements [Chandrupatla and Belegundu (2016)]. Therefore, a lot of time is spent in generation of well conditioned quadrilateral elements. The present work is restricted to two dimensional problems discretized with triangular elements; however, the solution accuracy is comparable to the quadrilateral elements. This is possible by using SFEM proposed by Liu et al. [Liu, Dai and Nguyen-Thoi (2007)]. Various methods proposed in the SFEM framework are the Cell Based SFEM (CSFEM), Node Based SFEM (NSFEM), Edge Based SFEM (ESFEM), Face Based SFEM (FSFEM), Selective SFEM, α-FEM, β-FEM and other variations [Zeng and Liu (2016)]. All the SFEM methods are applicable to triangular, quadrilateral or polygonal elements. These methods are well established with characteristics like convergence properties, stability, accuracy, and computational complexity [Nguyen-Xuan, Bordas and Nguyen-Dang (2008)]. Also the methods can be easily extended to three dimensional and geometric nonlinearity with nearly-incompressible material problems [Lee (2016)]. Kumbhar et al. [Kumbhar, Francis, Swaminathan et al. (2018); Wang (2014)], developed UEL to implement the CSFEM over the Polygonal Finite Element Method (PFEM) and the classical FEM in the commercial software Abaqus. Using CSFEM on two dimensional problems discretized with triangular elements, provides the numerical results exactly same as the classical FEM [Liu, Dai and Nguyen-Thoi (2007)]. The NSFEM is found to be suitable to work with triangular elements and less sensitive to mesh distortion and locking issues [Liu, Nguyen-Thoi, Nguyen-Xuan et al. (2009)].

In this work, Abaqus UEL is developed for the NSFEM and confined to the two dimensional problems. However this work can be easily extended to three dimensional problems. The problem domain is completely discretized using Constant Strain Triangular (CST) elements. The CST elements are chosen because the commercial software available can discretize any complex two dimensional geometry using CST elements without much difficulty [Altair Hypermesh Documentation (2014)]. Also the use of CST elements ease the construction of node based smoothing domain. Since, for the triangular elements, each node in the domain is associated with the one-third area of the corresponding attached element. All such regions for each node are known as smoothing regions. Therefore, the Abaqus UEL for NSFEM requires nodal coordinates and element connectivity of all the CST elements attached to each node under consideration. This data is provided to the Abaqus UEL through external data file. Depending upon the number of attached CST elements, the Abaqus UEL definition changes. For this purpose, a master UEL is written which incorporates different UEL elements defined as subroutines. The two dimensional linear and nonlinear elasticity problems are solved taking into account nearly-incompressible materials. For nonlinear large strain problems, hyperelastic material model, the quasi-compressible neo-Hookean model in this work, is used. The complete paper is divided in the following sections: the governing equations for the elasticity problems, NSFEM Abaqus implementation section explains the procedure involved for the NSFEM framework, the numerical examples section that validates the developed UEL by solving few benchmark problems and followed by the conclusion section.

2 Smoothed finite element approximation

In this section the governing equations are first presented for two dimensional linear elasticity and then its Smoothed finite element approximation is derived. In the subsequent subsection, the equations pertaining to weak form of two dimensional nonlinear elasticity accounting geometric nonlinearity are presented. The neo-Hookean hyperelastic material model and the Newton-Raphson iterative method equations for obtaining nonlinear solution are presented in brief.

2.1 Smoothed finite element approximation in linear elasticity

Let us consider a two dimensional isotropic linear elastic body in d ($\{\equiv 1, 2\}$) dimensional Euclidean space, $\mathbb{R}^d$, whose material point is given by $x = \sum x_I e_I$, where e_I are the vectors of a chosen basis. Let $\Omega \subset \mathbb{R}^d$ represent the body with domain boundary Γ. The body with domain boundary $\Gamma \equiv \delta\Omega$ consisting of Dirichlet boundary $\Gamma_{\mathbf{u}}$ and Neumann boundary $\Gamma_{\mathbf{t}}$, such that $\Gamma = \Gamma_{\mathbf{u}} \cup \Gamma_{\mathbf{t}}$ and $\Gamma_{\mathbf{u}} \cap \Gamma_{\mathbf{t}} = \emptyset$ and the outward normal to Γ is n. Given the body force $\mathbf{b}$, the Cauchy stress tensor $\boldsymbol{\sigma}$, the prescribed displacement $\bar{\mathbf{u}}$ and the traction $\bar{\mathbf{t}}$, to find the displacement field $\mathbf{u}$ the governing differential is written as below:

$$\nabla \cdot \boldsymbol{\sigma} \quad + \mathbf{b} = \mathbf{0} \quad in \quad \Omega, \tag{1a}$$

$$\mathbf{u} = \bar{\mathbf{u}} \ on \ \Gamma_{\mathbf{u}}, \tag{1b}$$

$$\boldsymbol{\sigma}.\mathbf{n} = \bar{\mathbf{t}} \ on \ \Gamma_{\mathbf{t}}, \tag{1c}$$

Let $\mathcal{U}(\Omega) = \{\mathbf{u}: \Omega \ \rightarrow \mathbb{R}^d \,|\, \mathbf{u}_I \in H^1(\Omega), I = 1,\ldots,d, \mathbf{u} = \bar{\mathbf{u}} \text{ on } \Gamma_{\mathbf{u}}\}$ be the displacement trial function and $\mathcal{V}(\Omega) = \{\mathbf{v}: \Omega \ \rightarrow \mathbb{R}^d | \mathbf{v}_I \in H^1(\Omega), I = 1,\ldots,d, \mathbf{v} = \mathbf{0} \text{ on } \Gamma_{\mathbf{u}}\}$ be the test function spaces, where H^1 denotes the Hilbert-Sobolev first order space. The variational form of Eq. (1a) is written as below:

$$\int_\Omega \boldsymbol{\sigma}.\nabla\mathbf{u} \ d\Omega = \int_\Omega \mathbf{b}.\mathbf{v} \ d\Omega + \int_{\Gamma_{\mathbf{t}}} \bar{\mathbf{t}}.\mathbf{v} \ d\Gamma \tag{2}$$

The Cauchy stress tensor $\boldsymbol{\sigma}$ and strain tensor $\boldsymbol{\varepsilon}$ can be expressed as:

$$\boldsymbol{\sigma} = 2\mu\boldsymbol{\varepsilon} + \lambda \text{tr}(\boldsymbol{\varepsilon})\mathbf{I} \tag{3}$$

where shear modulus μ and Lamé's first parameter λ can be expressed by Young's modulus E and Poisson's ratio ν as follows:

$$\mu = \frac{E}{2(1+\nu)}, \quad \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{4}$$

The strain tensor $\boldsymbol{\varepsilon}$ is given as:

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\Delta\mathbf{u} + \Delta\mathbf{u}^{\mathrm{T}}\right) \tag{5}$$

Note that, since the relation of the Cauchy stress tensor and strain tensor is given as $\boldsymbol{\sigma} = \mathbf{D}\boldsymbol{\varepsilon}$, the material modulus matrix $\mathbf{D}$ can be defined as:

$$\mathbf{D} = \begin{bmatrix} 2\nu + \lambda & \lambda & 0 \\ \lambda & 2\nu + \lambda & 0 \\ 0 & 0 & \lambda \end{bmatrix} \tag{6}$$

In the following, it is assumed that the arbitrary domain is partitioned into *nel* finite elements defined as ϖ_I such that $\Omega \equiv \sum_{I=1}^{nel} \varpi_I$ and $\varpi_I \cap \varpi_J = \emptyset, \forall I \neq J$. For the discretization of the weak form, let the set $\mathcal{U}(\Omega) \subset H^1$ consist of polynomial interpolation functions of the following form:

$$\mathbf{u}^{\boldsymbol{h}} = \sum_{e=1}^{nel} \phi_e \mathbf{u}_e$$
$$\mathbf{v}^{\boldsymbol{h}} = \sum_{e=1}^{nel} \phi_e \mathbf{v}_e \tag{7}$$

where ϕ_e denote the finite element shape functions. Substituting these displacement functions in Eq. (2), the corresponding weak form can be obtained as:

$$\int_\Omega \mathbf{D}\boldsymbol{\epsilon}(\mathbf{u}^h)\boldsymbol{\epsilon}(\mathbf{v}^h)\ d\Omega = \int_\Omega \mathbf{b}.\mathbf{v}^h\ d\Omega + \int_{\Gamma_\mathbf{t}} \bar{\mathbf{t}}.\mathbf{v}^h\ d\Gamma \tag{8}$$

Using the arbitrariness of testing displacement function $\mathbf{v}^h$, this can be reduced to the following form:

$$\mathbf{Ku} = \mathbf{f}$$
$$\mathbf{K} = \sum_h \mathbf{K}^h = \sum_h \int_{\Omega^h} \mathbf{B}^\mathbf{T}\mathbf{DB}\, d\Omega$$
$$\mathbf{f} = \sum_h \mathbf{f}^h = \sum_h \left(\int_{\Omega^h} \phi^T\mathbf{b}\, d\Omega\ + \int_{\Gamma^h} \phi^T\bar{\mathbf{t}}\, d\Gamma\right) \tag{9}$$

In Eq. (9), $\mathbf{K}$ is the global stiffness matrix and $\mathbf{B} = \nabla\phi$ is the strain-displacement matrix that is computed using the derivatives of the shape functions ϕ for each element in case of the classical FEM. Liu et al. [Liu, Dai and Nguyen-Thoi (2007a)] employed SFEM converting area integration to line integration through the divergence theorem while performing numerical integration. The basic idea of SFEM is to divide the problem domain into sub-domains where strains are smoothed. These strains are constant over the smoothing domains but they are discontinuous across the boundaries of these smoothing domains. The smoothing domains are constructed using the topology provided by the mesh. For a single CST element shown in Fig. 1(a), NSFEM smoothing domains for respective nodes are marked with distinct features.

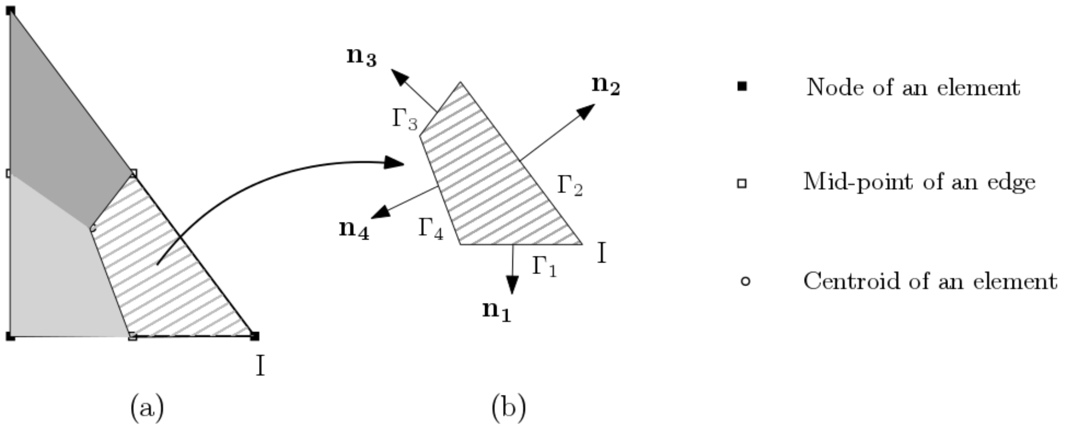


Figure 1: NSFEM smoothing domains for a CST element

The smoothed strain-displacement matrix $\mathbf{B_I}$ for node I is evaluated as below:

$$\mathbf{B_I} = \frac{1}{A_I}\int_\Gamma \mathbf{n}\phi_I{}^T\, d\Gamma \tag{10}$$

where A_I is the area of the smoothing domain, $\mathbf{n}$ is the outward normal vector and Ψ_I is the shape functions. Referring to Fig. 1(b), it is observed that overall numerical integration consists of line integration over the smoothing domain boundaries in SFEM compare to classical FEM. Such a line integration is evaluated using the shape function value at one Gauss point located at mid-point of the line boundary and the corresponding outward normal. After the evaluation of the strain-displacement matrix, construction of global stiffness matrix $\mathbf{K}$ is similar to classical FEM. Thus in SFEM numerical integration is dependent on node coordinates in physical space avoiding locking and element distortion issues associated with classical FEM. The detailed construction and simplified evaluation of the strain displacement matrix for NSFEM having more than one CST elements will be explained in Section 3.1.

2.2 Smoothed finite element approximation in finite elasticity.

Here for the completeness of the subsection, few important equations and outline of the nonlinear elasticity solution is presented. However the elaborate derivations and detailed discussions of nonlinear elasticity accounting geometric nonlinearity and nearly incompressible material are found in Bhat [Bhatti (2006)].

Galerkin variational form for finite elasticity can be written as:

$$\int_\Omega \frac{\partial W}{\partial F}(X, F(\mathbf{u})) : \nabla\mathbf{v}\ d\Omega = \int_\Omega \mathbf{b}.\mathbf{v}\ d\Omega + \int_{\Gamma_t} \bar{\mathbf{t}}.\mathbf{v}\ d\Gamma \tag{11}$$

where W is the strain energy density.

In this work, the compressible neo-Hookean model is used [Bhatti (2006)]:

$$W = \frac{1}{2}\lambda(\ln J)^2 - \mu\ln J + \frac{1}{2}\mu(\mathrm{tr}\boldsymbol{C} - 3) \tag{12}$$

where the right Cauchy-Green tensor $\boldsymbol{C}$ and the Jacobian J are defined as $\boldsymbol{C} = F^T F$ and $J = \det F$ respectively. The deformation gradient F can be evaluated as:

$$F_{ij} = \left(\frac{\partial x_i}{\partial X_j}\right) \tag{13}$$

where X is the initial configuration and x is the current configuration expressed as $x = X + \mathbf{u}$.

Note that, the left hand side and the right hand side of the Eq. (11) are defined as internal forces and external forces respectively:

$$\begin{aligned} \mathbb{W}_{int} &= \int_\Omega \frac{\partial W}{\partial F}(X, F(\mathbf{u})) : \nabla\mathbf{v}\ d\Omega \\ \mathbb{W}_{ext} &= \int_\Omega \mathbf{b}.\mathbf{v}\ d\Omega + \int_{\Gamma_t} \bar{\mathbf{t}}.\mathbf{v}\ d\Gamma \end{aligned} \tag{14}$$

The Newton-Raphson iterative method is used for the estimation of nonlinear solution. In this method, initial solution is assumed as zero and a residual is generated in Eq. (11). Subsequent incremental solutions are estimated using directional derivative of $\mathbb{W}_{int}$ and $\mathbb{W}_{ext}$. The equation for residual $R_\mathbf{u}$ with the current displacement function $\mathbf{u}$ can be written as:

$$R_\mathbf{u} = \mathbb{W}_{int} - \mathbb{W}_{ext} \tag{15}$$

To estimate incremental iterative displacement $\Delta\mathbf{u}$, Eq. (13) can be expressed with the directional derivative of $\mathbb{W}_{int}$ and $\mathbb{W}_{ext}$ as follows:

$$\mathbb{W}_{int} + D_{\Delta\mathbf{u}}\mathbb{W}_{int} = \mathbb{W}_{ext} + D_{\Delta\mathbf{u}}\mathbb{W}_{ext} \tag{16}$$

where the directional derivative for work done by internal forces $D_{\Delta\mathbf{u}}\mathbb{W}_{int}$ is given as:

$$D_{\Delta\mathbf{u}}\mathbb{W}_{int} = \int_{\Omega} \frac{\partial^2 W}{\partial F_{ij}\partial F_{kl}}\big(X, F(\mathbf{u})\big)\frac{\partial v_k}{\partial X_l}\frac{\partial v_i}{\partial X_j}d\Omega \tag{17}$$

where $i, j, k, l \in (1,2)$ for two-dimensions. Note that $D_{\Delta\mathbf{u}}\mathbb{W}_{ext} = 0$ as external forces will be unaffected by displacement changes except external pressure forces which are deformation dependent.

Combining Eq. (16) and Eq. (17) gives a system of linear equations. Solution of this system of linear equations leads to the estimation of displacement increment $\Delta\mathbf{u}$. With the new displacement $\mathbf{u}_{iter+1} = \mathbf{u}_{iter} + \Delta\mathbf{u}_{iter}$, residual $R_{\mathbf{u}}$ is recalculated. To get more accurate solutions, the residual is required to be minimum.

3 NSFEM Abaqus implementation

3.1 Construction of node based smoothing domain in the framework of Abaqus UEL definition

Fig. 2 shows an example of the construction of node-based smoothing domain Ω_c. As shown in Fig. 2, when node c is selected as the target node, neighboring CST elements are divided into subcells by mid-points and centroids. Each neighbored subcell with vertices labelled, e.g., a1,c, a11 and a12, is connected to anti-clockwise manner. When connected subcells make a closed region, it is called as the smoothing domain.The strain-displacement matrix $\boldsymbol{B_{NR}}$ of the smoothing domain Ω_c can be evaluated as follows [Liu, Nguyen-Thoi, Nguyen-Xuan et al. (2009)]:

$$\boldsymbol{B_{NR}} = \frac{1}{\boldsymbol{A_{NR}}}\sum_{n=1}^{N}\frac{1}{3}\boldsymbol{A_n}\boldsymbol{B_n} \tag{18}$$

where $\boldsymbol{A_{NR}}$ is the area of the smoothing domain Ω_c, N is the number of CST elements sharing node c, $\boldsymbol{A_n}$ is the area of CST element and $\boldsymbol{B_n}$ is the strain-displacement matrix of CST element in FEM. Since the compatible strain is constant in CST element, simplified node-based strain-displacement matrix $\boldsymbol{B_{NR}}$ (Eq. (20)) can be used in the proposed Abaqus UEL.

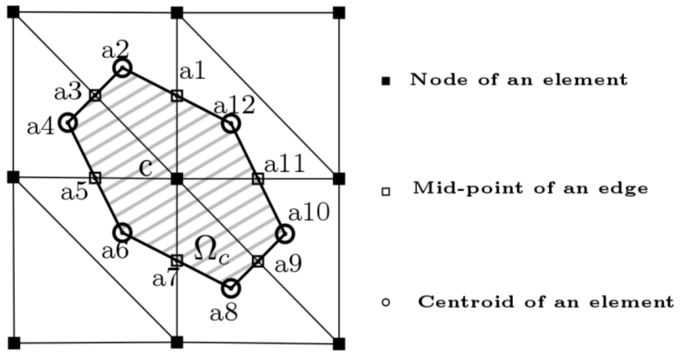


Figure 2: Node-based smoothing domain associated with a target node

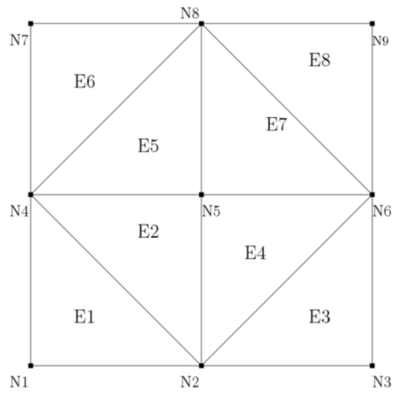


Figure 3: Discretization of a square domain by triangular meshes with nodes labelled N1 to N8 and elements labelled E1 to E8

```
U1 UEL for Node 1 will be defined as:
*User element, nodes=3, type=U1, properties=3, coordinates=2, variables=7
1,2
*Element, type=U1, Elset=COMP
1,1,2,4
U4 UEL for Node 2 will be defined as:
*User element, nodes=6, type=U4, properties=3, coordinates=2, variables=7
1,2
*Element, type=U4, Elset=COMP
2,2,3,6,5,4,1
U3 UEL for Node 5 will be defined as:
*User element, nodes=5, type=U3, properties=3, coordinates=2, variables=7
1,2
*Element, type=U3, Elset=COMP
5,5,6,8,4,2
```

Listing 1: Definition of different NSFEM Abaqus UEL elements

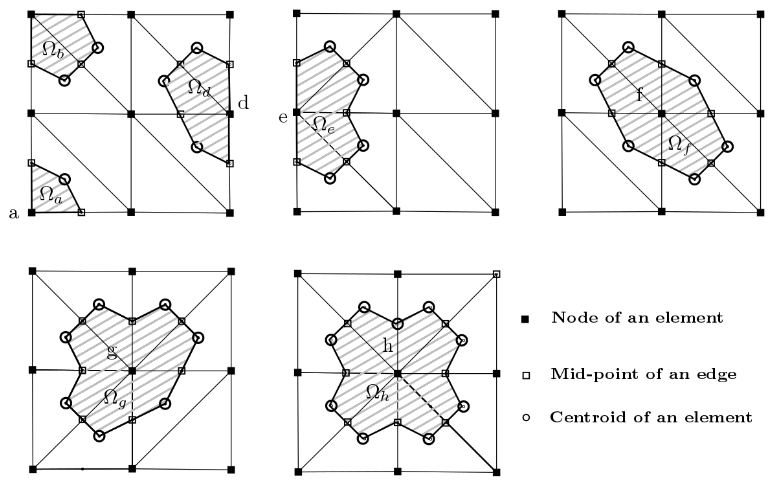


Figure 4: Smoothing domains for the NSFEM Abaqus UEL elements U1 to U7

For the NSFEM Abaqus UEL definition, first let discretize a square domain with triangular meshes as shown in Fig. 3. Then it needs to be found element cells associated with each node, for example, Node N1 is contained to node vertices of element E1 while node N2 is attached to elements E1 to E4. This means that targeted node N1 is attached to only element E1 and node vertices of element E1 are N1, N2 and N4. Similarly, for node N2, attached elements have node vertices N1, N2, N3, N4, N5 and N6. This node information for the proposed UEL definition is given in Listing 1, sampling node N1, N2 and N5 with prefix N dropped. The different NSFEM UEL element numbering like U1, U2, U3, etc. is used according to the number of nodes of attached elements. The number of variables in the UEL is chosen according to analysis requirements. It can be also observed in Listing 1 that the number of nodes in the NSFEM Abaqus UEL is equal to UEL specific number after letter U plus two. In addition, similar property Abaqus UEL elements can be grouped together and can be given a particular name for easy reference. In the same manner, different definition types of smoothing domains corresponding to the NSFEM Abaqus UEL are given in Fig. 4.

3.2 Major steps in the NSFEM Abaqus UEL implementation

Fig. 5 depicts the three major steps involved in the implementation of the NSFEM Abaqus UEL: 1) Pre-processing, 2) Analysis using NSFEM Abaqus UEL and 3) Post-processing.

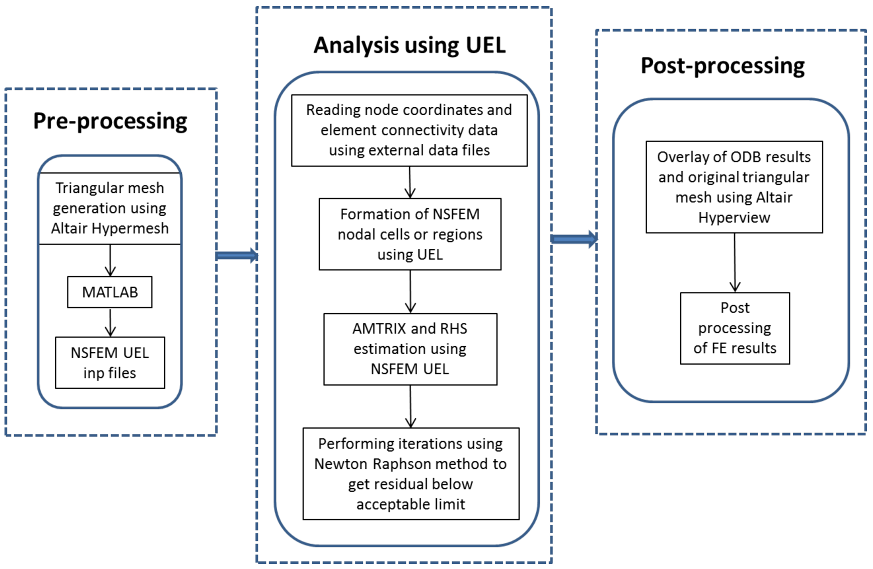

Pre-processing
Triangular mesh generation using Altair Hypermesh
MATLAB
NSFEM UEL inp files
Analysis using UEL
Reading node coordinates and element connectivity data using external data files
Formation of NSFEM nodal cells or regions using UEL
AMTRIX and RHS estimation using NSFEM UEL
Performing iterations using Newton Raphson method to get residual below acceptable limit
Post-processing
Overlay of ODB results and original triangular mesh using Altair Hyperview
Post processing of FE results

Figure 5: Major steps involved in the implementation of NSFEM Abaqus UEL

3.2.1 Pre-processing

The first stage of the proposed UEL element is the pre-processing where the problem domain is discretized using CST elements. For the domain discretization, the mesh generation software Altair Hypermesh [Altair Hypermesh Documentation (2014)] is used. NSFEM Abaqus UEL is defined using the nodes of attached elements as explained in the previous section. A MATLAB code is used to obtain node vertices of elements associated with the considering node. Once the attached elements are found, their node vertices are arranged after the target node to define NSFEM Abaqus UEL. Data set of node coordinates and node connectivity of CST elements is used as input parameters for the MATLAB code. The data files of the NSFEM Abaqus UEL definition are automatically generated using the MATLAB code.

Before proceeding towards the analysis stage, input data file compatible with the developed UEL requires to share the following common keywords [Abaqus Documentation (2012)]: *user element to define the UEL and *UEL property to define thickness and material properties for element groups. In the keyword of *user element, the NSFEM Abaqus UEL elements labelled like U1, U2, U3, etc. need to be matched appropriately with the corresponding element type in the Abaqus input data file. Moreover, the NSFEM Abaqus UEL elements having the same number of nodes can be grouped into one and given a specific name. Rest of the structure of input data file is similar to the typical Abaqus standard input data file.

3.2.2 Analysis using SFEM Abaqus UEL

The next stage is to run the analysis using NSFEM Abaqus UEL. For the analysis Abaqus standard solver can be invoked by the following command:

```
abaqus job=<inp file name>  user=<fortran Abaqus UEL file name> other options
```

This command is much similar to the one used for other types of Abaqus UEL. The difference with NSFEM Abaqus UEL is that the data file containing the data set of CST element node coordinates and node connectivity is provided during the analysis. Such data file is kept in the same file path as the other analysis files. Note that, this data file path and its name need to be carefully mentioned in the definition of the NSFEM master UEL. In this stage use of the NSFEM Abaqus UEL FORTRAN code for the analysis is included as well.

3.2.3 Post-processing

Once the analysis is completed, Abaqus output data base (ODB) file containing the requested result is generated in the last stage. Since Abaqus CAE does not support post-processing of Abaqus UEL elements without significant modifications, Altair Hyperview [Altair Hypermesh documentation (2014)], in the present work, is used for the post-processing. In this case Abaqus ODB results file and Abaqus input data file containing CST elements are overlaid for the post-processing.

3.2.4 The details of the NSFEM Abaqus UEL FORTRAN code

To complete the implementation of the NSFEM, following FORTRAN code for Abaqus UEL is required. In this work, the NSFEM Abaqus UEL structure is arranged as a master UEL which further calls an individual UEL like U1, U2, etc. as a subroutine. The outline of the master UEL is shown in Listing 2. The master UEL starts with the name of the subroutine with the list of arguments in the parentheses. Parameters section includes names of the constants which are assigned a fixed value. This leads the identification of constants easy while using them in the FORTRAN code. RHS and AMATRIX matrices denote the residual force and stiffness contribution of each element respectively are also completely defined in the UEL. Additional variables such as numnode, numelem, node and element are added in the UEL to read node coordinates and elements connectivity data from an external data file as shown in Listing 2.

The structure of the data file contains total number of nodes at the top and followed by node coordinates. It is ensured that all the nodes are renumbered. Using the node numbers, their coordinates are arranged in ascending order. In this work, node numbers can be eliminated as sequence itself dictates the node number. If the renumbering of nodes is not done, it is necessary to include the node numbers in the data file. After the node coordinates in the data file, total number of CST elements in the domain are written and this is followed by node connectivity data of the CST elements. Similar to the node renumbering, the same comments are applied to the element renumbering. The if statement in the NSFEM master UEL is used to select the corresponding UEL subroutine according to the NSFEM Abaqus UEL element type.

A detailed definition of a UEL U1 is shown in Listing 3. In this detailed code, it can be observed that element number of NSFEM Abaqus UEL which is actually the node for which the smoothing domain is defined. This node number is used for finding the attached CST elements. Eq. (20) is used for the calculation of thestrain-displacement matrix constructed over the NSFEM smoothing domain. The detailed subroutine definition helps in the estimation of RHS and AMATRIX of the corresponding NSFEM smoothing domain. The estimation of global stiffness matrix by assembling of the matrices of individual UEL elements is taken care by Abaqus.

The Newton-Raphson iterative method is used for correcting the nonlinear stiffness of the model in the same manner with the conventional FEM. Abaqus handles the iterative method once the linear and nonlinear stiffness matrices are defined correctly in the NSFEM Abaqus UEL subroutine. Material properties are read by the keyword *UEL property from the Abaqus input data file. Iterations are performed till it satisfies the convergence criteria as set in the Abaqus analysis parameters. Overall the major difference in the analysis using NSFEM Abaqus UEL is the estimation of the node-based strain-displacement matrix over smoothing domains but the rest of the process is the same as FEM[4].

[4]All MATLAB and FORTRAN codes are downloadable and available on Github (hppts://github.com/nsundar/NSFEM)

```
C  User subroutine for NSFEM method with base mesh of CST elements
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
1     PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
2     KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,
3     NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,
4     PERIOD)
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
                INCLUDE 'ABA_PARAM.INC'
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
      parameter(zero=0.d0, half=0.5, one=1.d0, two=2.d0, three=3.d0,
    1 four=4.d0, six=6.d0, eight=8.d0, twelve=12.d0)
C
INTEGER :: numnode,numelem
REAL(8), DIMENSION(:,:), ALLOCATABLE :: node
INTEGER, DIMENSION(:,:), ALLOCATABLE :: element
C
      open(10,file='File path for reading  triangular mesh file data')
     read(10,*) numnode
     if(.not.allocated(node)) allocate(node(numnode,2))
     do i=1,numnode
     read(10,*) (node(i,j),j=1,2)
     enddo
     read(10,*) numelem
     if(.not.allocated(element)) allocate(element(numelem,3))
     do i=1,numelem
     read(10,*) (element(i,j),j=1,3)
     enddo
     close(10)
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
C numnode and numelem are the total number of nodes and elements in the
C domain. node and element are matrices containing node coordinates and
C element nodes. The above quantities are assigned values using external
C data file mesh.dat
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
C Use of if statement to choose particular UEL like U1, U2, U3 etc.
     if(jtype.eq.1) then
C This is for a single node(may be at the corner of global mesh) having three surounding
nodes
C (including the node itself) or one adjacent element
        call U1(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
                1     PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
                2     KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,
                3     NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,
                4     PERIOD,numnode,numelem,node,element)
     elseif(jtype.eq.2) then
C This is for a node on the boundries or corners or inside the global mesh surrounded by
four
C nodes or having more than one adjacent elements
        call U2(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
                1     PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
                2     KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,
                3     NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,
                4     PERIOD,numnode,numelem,node,element)
                elseif(jtype.eq.3) then
C Like this all the user elements like U1,U2,U3 etc. used in the input file are
C defined.
 else
C
        write(*,*) 'Element type not supported, jtype=',jtype
        write(80,*) 'Element type not supported, jtype=',jtype
        call exit
C
     endif
     return
     end subroutine UEL
C This is followed by definition of each UEL in detail
C One can request printing of any of the variables using write statement
C This is the main structure of NSFEM UEL
```

Listing 2: NSFEM master UEL FORTRAN code outline

```
C Definition of an individual UEL
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
SUBROUTINE U1(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
1     PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
2     KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,
3     NPREDF,LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,
4    PERIOD,numnode,numelem,node,element)
•  Numnode, numelem, node and element variables are used for the
   estimation of strain displacement matrix of CST elements.
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
               INCLUDE 'ABA_PARAM.INC'
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
DIMENSION RHS(MLVARX,*),AMATRX(NDOFEL,NDOFEL),
1     SVARS(NSVARS),ENERGY(8),PROPS(*),COORDS(MCRD,NNODE),
2     U(NDOFEL),DU(MLVARX,*),V(NDOFEL),A(NDOFEL),TIME(2),
3     PARAMS(3),JDLTYP(MDLOAD,*),ADLMAG(MDLOAD,*),
4     DDLMAG(MDLOAD,*),PREDEF(2,NPREDF,NNODE),LFLAGS(*),
5     JPROPS(*)
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
C-------
C---------- Define parameters, dimensions and element type
•  Read material property data for group of elements
•  For first iteration, quantities like displacements, strains, stresses
   etc. are set to zero.
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
C---------- Initialize matrices
* Initialization of AMATRIX and RHS to zero.
C---------- Define coordinates for N1-N3--------C-------C-------C-------C-------
* Using element number (node number) find the number of attached
triangular elements. By knowing attached elements, NSFEM UEL
strain displacement matrix can be calculated.
C---------- Assemble Constitutive matrix---D matrix-----C-------C-------C-
Evaluate stiffness matrices related to geometric non-linearity. Details
of the code can be found in external files.
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
 *  Finally evaluate RHS and AMATRIX. Details of code found in
    external files.
 *  For geometric non linear problem, Abaqus solver will repeat the
    procedure till the residual is within acceptable limit.
C-------C-------C-------C-------C-------C-------C-------C-------C-------C-------
return
end subroutine U1
```

Listing 3: FORTRAN code outline for the NSFEM Abaqus UEL U1

4 Numerical examples

In this section, a series of numerical examples is solved to demonstrate the utility of Abaqus NSFEM UEL elements: a cantilever beam, a bevelled cantilever beam, a square plate with a hole and a sharp V-notched square plate. For the tests, two dimensional linear elasticity reference problem is taken from Liu et al. [Liu, Dai and Nguyen-Thoi (2007)] and nonlinear elasticity considering geometric nonlinearity with nearly-incompressible material reference problems is chosen from Lee [Lee (2016)]. Obtained numerical results are compared with Abaqus results with quadrilateral elements.

The accuracy and the convergence rate using different Abaqus elements are analyzed using the relative error norm in the displacement and strain energy as given by:

Displacement norm:

$$\| \mathbf{u} - \mathbf{u}^h \|_{L^2(\Omega)} = \frac{\sqrt{\int_\Omega (\mathbf{u}-\mathbf{u}^h)^T(\mathbf{u}-\mathbf{u}^h) d\Omega}}{\sqrt{\int_\Omega \mathbf{u}^T\mathbf{u} \; d\Omega}} \tag{19}$$

Strain energy norm:

$$\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}^h \|_{H^1(\Omega)} = \frac{\sqrt{\int_\Omega (\boldsymbol{\epsilon}-\boldsymbol{\epsilon}^h)^T D(\boldsymbol{\epsilon}-\boldsymbol{\epsilon}^h) d\Omega}}{\sqrt{\int_\Omega \boldsymbol{\epsilon}^T D \boldsymbol{\epsilon} \; d\Omega}} \tag{20}$$

4.1 Patch test

In this section, two patch tests are performed on arbitrary patches of elements to satisfy basic convergence requirements of rigid body displacements and constant strain conditions to validate Abaqus NSFEM UEL. Note that, the material properties are Young's modulus $E = 1000.0$ Pa and Poisson's ratio $\nu = 0.3$ and only pre-described displacement boundary conditions are considered for the tests.

Firstly, a rigid body motion displacement is considered with an arbitrary patch of CST elements as shown in Fig. 6(a). The pre-described horizontal displacements which is 1.0 m are imposed on boundary nodes a1, a2, a3, a4, a7, a9 and a10. In this test, as a result, the computed horizontal displacements at interior nodes a5, a6 and a8 should be 1.0 m. As shown in Fig. 6(b), obtained results show that Abaqus NSFEM UEL passes the test.

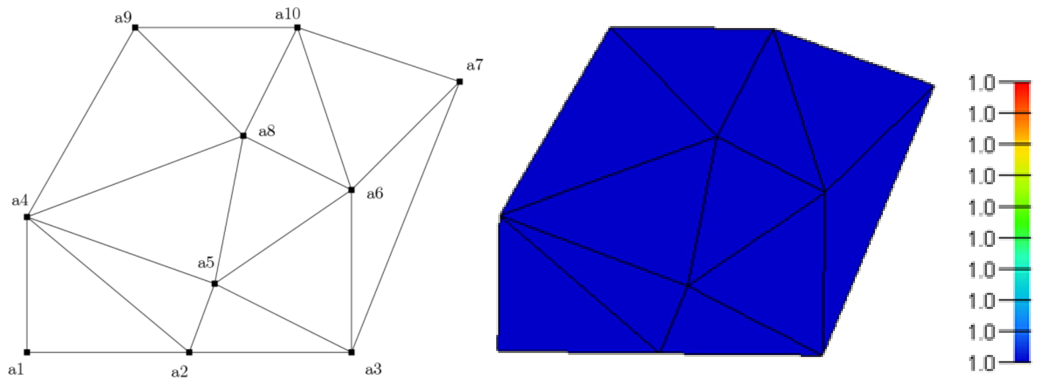


Figure 6: Rigid body motion displacement test with CST elements: (a) geometry of the patch and (b) constant strain displacement results

In the next test, linear displacements $\mathbf{u} = x$ are imposed on the boundary nodes b1, b2, b3, b4, b6, b7, b8 and b9 for the patch as shown in Fig. 7(a). In order to pass this constant strain patch test, interior node b5 must show horizontal displacement equal to its x coordinate. As shown in Fig. 7(b), Abaqus NSFEM UEL also passes the constant strain displacement test.

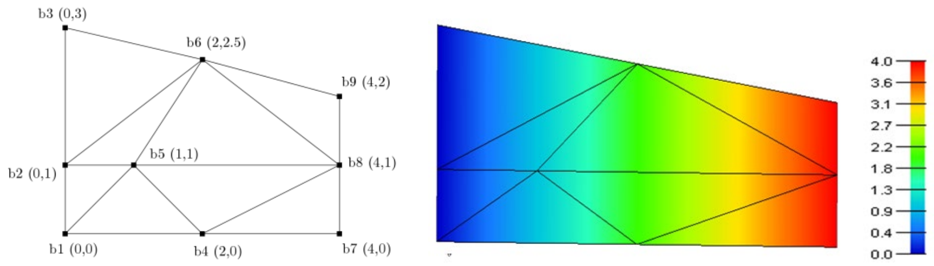


Figure 7: Constant strain displacement test with CST elements: (a) geometry of the patch and (b) constant strain displacement results

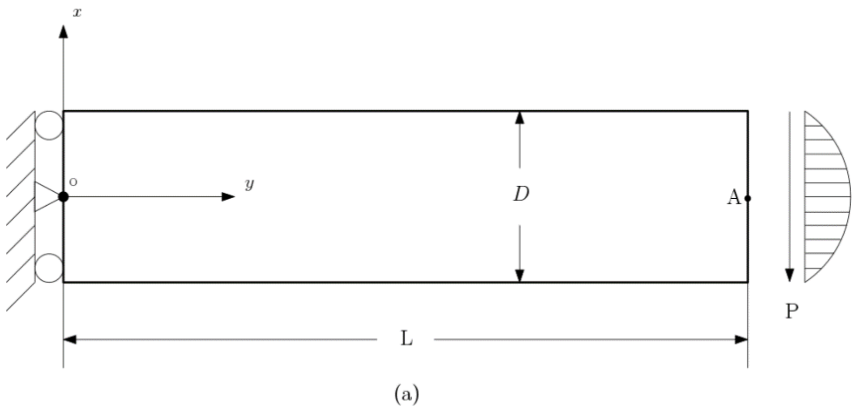
4.2 Cantilever beam in linear elasticity

The geometry of a cantilever beam with length $L = 48.0$ m and width $D = 12.0$ m and CST element discretization are shown in Fig. 8. The material parameters for the plate are $E = 3.0 \times 10^7$ Pa, $\nu = 0.3$ and the total parabolic shear load acting over the free edge is $P = -1000$ N.

The exact analytical solution for the above problem is given by:

$$\begin{aligned} u(x,y) &= \frac{Py}{6EI}\left[(6L-3x)x + (2+\nu)\left(y^2 - \frac{D^2}{4}\right)\right] \\ v(x,y) &= -\frac{P}{6EI}\left[3\nu y^2(L-x) + (4+5\nu)\frac{D^2 x}{4} + (3L-x)x^2\right] \end{aligned} \tag{21}$$

where $I = D^3/12$ is the moment of inertia and a state of plane stress is considered.



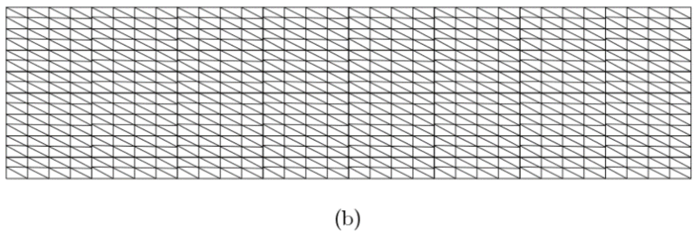
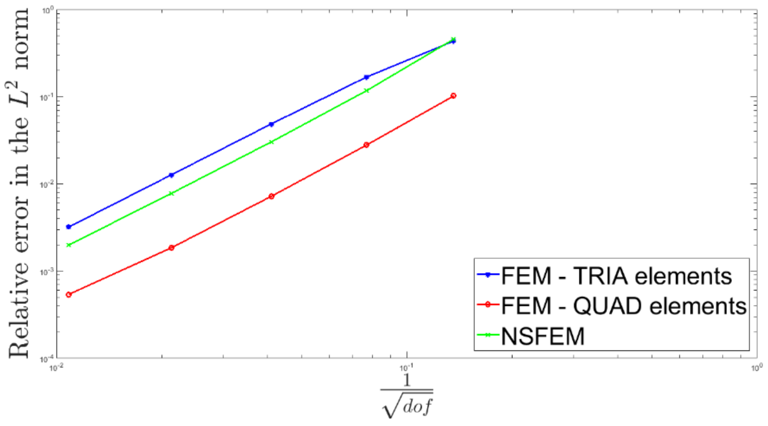


Figure 8: The geometry of a cantilever beam: (a) geometry and boundary conditions and (b) discretization of the beam with CST elements

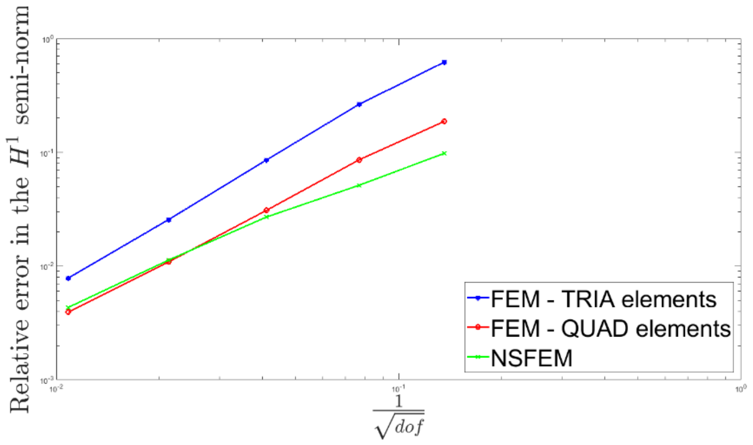
The convergence rates of the relative error in displacement norm and strain energy norm are given in Fig. 9. It is clearly seen that Abaqus NSFEM UEL shows better results than that using Abaqus triangular elements without any change in the domain discretization. These results are encouraging in the sense that analysis with NSFEM Abaqus UEL elements provide comparable results as those obtained using Abaqus quadrilateral elements with the same solver. In the next subsections, two dimensional nonlinear elasticity problems are considered.

4.3 Cantilever beam in geometric nonlinearity

In this section, the geometric nonlinearity is investigated. For this test, the cantilever beam used in the previous section is considered again. The parabolic shear load 10,000 N is implemented on the right-end edge and the left-end edge of the beam is completely constrained in all DOFs. The reference solution for this test is obtained by Abaqus with the very fine meshes (526,850 DOFs).



(a) L^2 norm



(b) H^1 norm

Figure 9: The convergence of the relative error for the cantilever beam: (a) L^2 norm and (b) H^1 semi-norm

Fig. 10 illustrates the deformed shapes of the beam for Abaqus QUAD CPS4I and the proposed Abaqus NSFEM UEL elements and an improved accuracy obtained by the proposed elements can be observed in Tab. 1.

The L^2 error norm convergence rate for this test is shown in Fig. 11. NSFEM UEL elements result shows better convergence rate than triangular elements and it is comparable to quadrilateral elements as shown in Fig. 11.

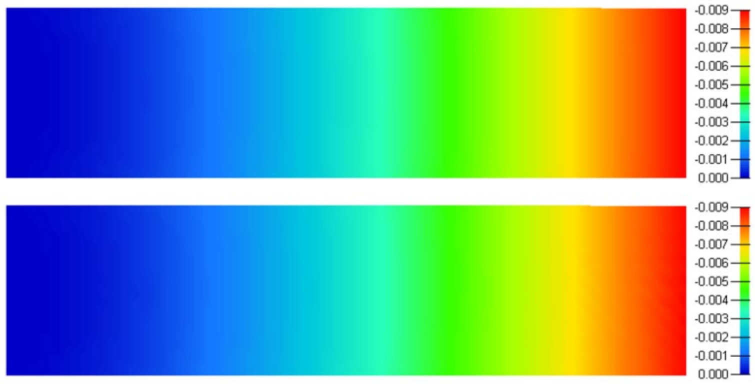


Figure 10: Deformed shapes of the cantilever beam with the vertical displacement plot: (a) Abaqus QUAD CPS4I and (b) Abaqus NSFE UEL elements

Table 1: The vertical displacements at the sampling point 'A' of the cantilever beam

Reference solution: -0.08907 (m)

Mesh (DOFs)	QUAD CPS4I	NSFEM UEL
54	-0.0879805	-0.1293490
170	-0.0885215	-0.0985716
594	-0.0888464	-0.0909835
2210	-0.0889910	-0.0890785

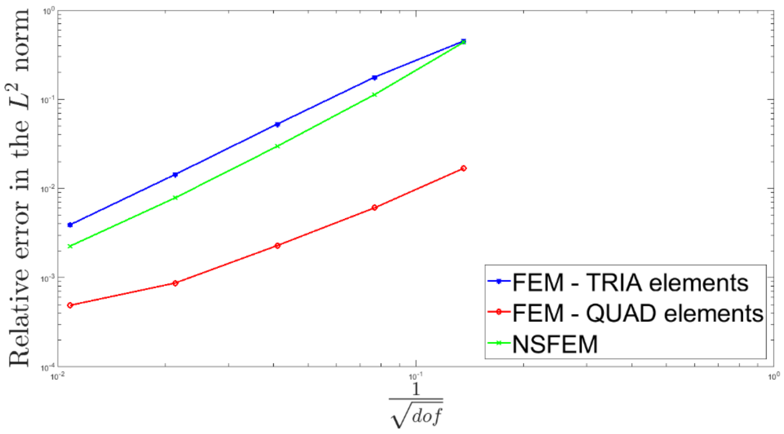


Figure 11: The convergence of the relative error in the L^2 norm for the cantilever beam

4.4 Bevelled Cantilever beam

A bevelled cantilever beam is studied in this section for the quasi-incompressible hyperelasticity. The geometry of the beam is given in Fig. 12 with $L1 = 0.5$ m and vertical load $P1 = -0.1$ N/m. In this work, the neo-Hookean model is used with shear modulus $\mu = 0.6$ Pa and bulk modulus $\kappa = 10^7$Pa equivalent to Poisson's ratio $\nu = 0.49999997$. Abaqus with very fine meshes (526,338 DOFs) is also used as the reference solution for this problem.

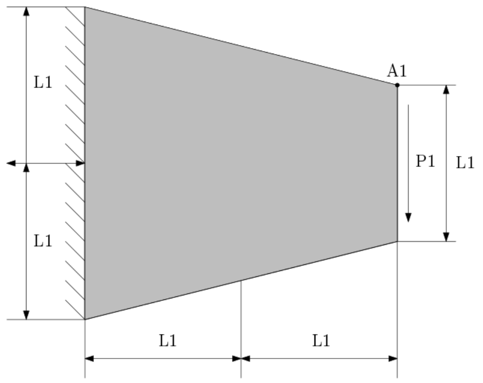


Figure 12: The geometry of a bevelled cantilever beam

The deformed shapes of the bevelled beam for Abaquas CPE4I and Abaqus NSFEM UEL elements are shown in Fig. 13 and their detailed displacement values at the sampling point 'A1' are given in Tab. 2. The results solved using NSFEM Abaqus UEL are in agreement with the results provided in Lee [Lee (2016)].

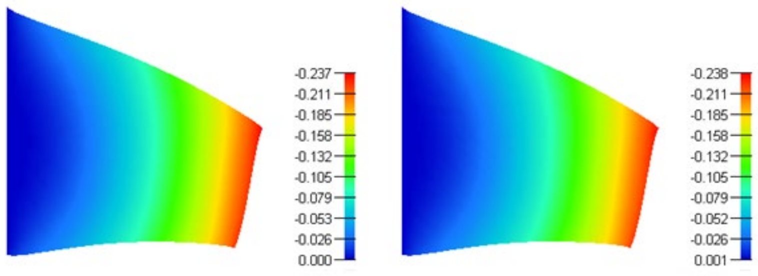


Figure 13: Deformed shapes of the bevelled cantilever beam: (a) Abaqus quadrilateral CPE4I and (b) Abaqus NSFEM UEL elements

Table 2: The vertical displacements at the sampling point 'A1' of the bevelled beam

Mesh (DOFs)	QUAD CPS4I elements (m)	NSFEM UEL elements (m)
18	-0.193628	-0.158183
50	-0.225601	-0.217090
162	-0.233652	-0.239422
578	-0.235873	-0.239782
2178	-0.236849	-0.237986

The convergence of the L^2 error norm is depicted in Fig. 14. It is observed from Fig. 14 that Abaqus NSFEM UEL provides better convergence rate than Abaqus triangular element. In addition the following displacement and convergence results are confirmed again: the proposed Abaqus UEL is comparable to fully integrated elements.

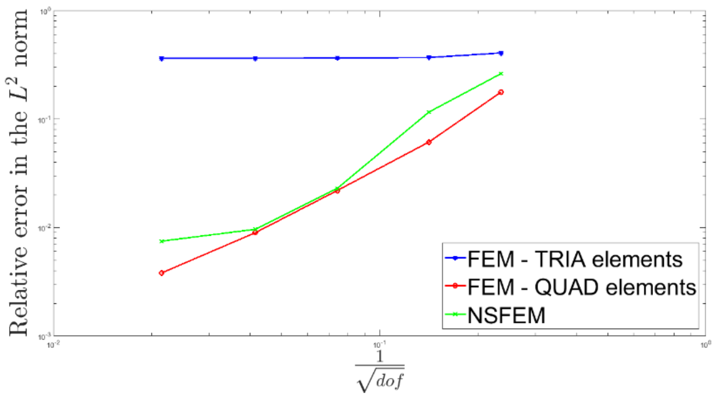


Figure 14: The convergence of the relative error in the L^2 norm for the bevelled cantilever beam

4.5 A square plate with a hole

In this section, nonlinear nearly-incompressible problem is once again demonstrated. The problem domain is given as a square plate with a hole at the center (see Fig. 15).

The geometry of the plate is $L2 = 1.0$ m and the radius of the circle is given as $R = 0.5$ m. Vertical external force $P2 = -0.1$ N is equally distributed on the top edge of the plate. The neo-Hookean material with Lamé's parameters $\mu = 1.8$ Pa and $\kappa = 10^7$ Pa are used (Poisson's ratio $\nu = 0.49999997$). Right, left and bottom edges are fully constrained in all DOFs.

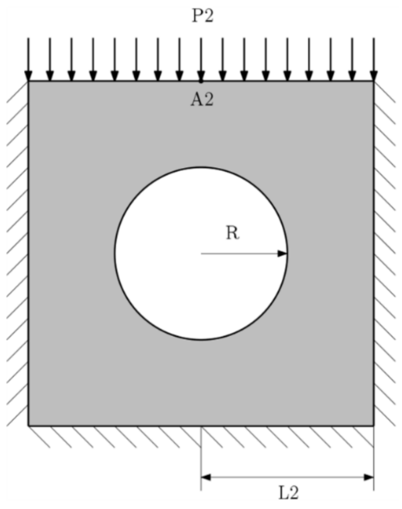


Figure 15: The geometry of a square plate with a hole at the center

The reference solution to the square plate with a hole problem is obtained using Abaqus with fine CPE4I element meshes having 791,712 DOFs. Fig. 16 shows deformed shapes of the square plate for Abaqus quadrilateral and the proposed NSFEM UEL elements. A comparison of vertical displacements of Abaqus CPE4I and Abaqus NSFEM UEL elements is given in Tab. 3 with detailed values. The results of vertical displacement obtained at the sample point 'A2' show an upper bound solution using NSFEM Abaqus UEL elements while that Abaqus CPE4I elements give a lower bound solution.

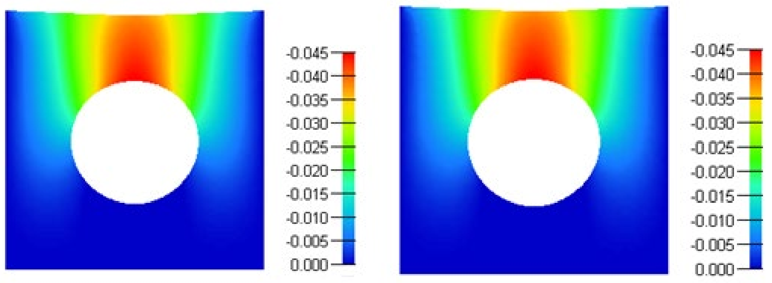


Figure 16: The deformed shapes of the plate: (a) Abaqus quadrilateral CPE4I and (b) the proposed Abaqus NSFEM UEL

Table 3: The vertical displacements at the sampling point 'A2' of the square plate

Reference solution: -0.0349782 (m)

Mesh (DOFs)	QUAD CPE4I elements (m)	NSFEM UEL elements (m)
218	-0.0365862	-0.0424276
1214	-0.0367179	-0.0383165
3740	-0.0374940	-0.0380560

4.6 A sharp V-notched square plate

Lastly, a square plate with a sharp V-notch is studied. The geometry of the plate is given as $L3 = 1.0$ m and $B = 0.02$ m as shown in Fig. 17. The loading parameter is given as the total vertical load $P3 = 0.05$ N on the top edge and shear modulus $\mu = 0.6$ Pa and bulk modulus $\kappa = 10^5$ Pa are used ($\nu = 0.499997$). The bottom edge of the plate is constrained in all DOFs and the vertical displacement results are reported at the point 'A3'.

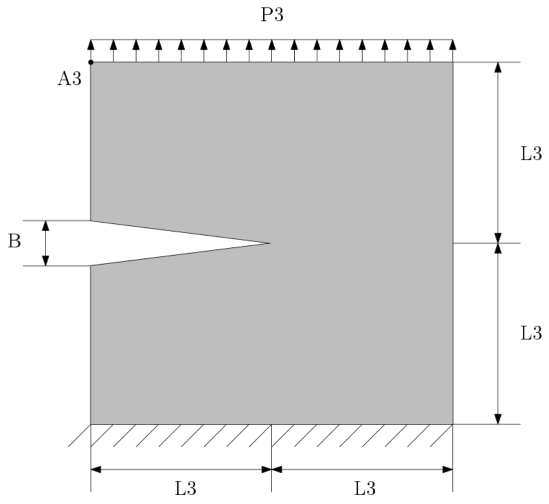


Figure 17: The geometry of a square plate with a sharp V-notch

Abaqus CPE4I element with the fine meshes (526,850 DOFs) is used as the reference solution for this test. The deformed shapes of Abaqus CPE4I and Abaqus NSFEM UEL for the plate are illustrated in Fig. 18. The detailed vertical displacements obtained at the sampling point 'A3' can be found in Tab. 4. Similar to the previous plate with a hole problem, the proposed UEL elements provide the upper-bound solution whereas Abaqus CPE4I results lower-bound solution.

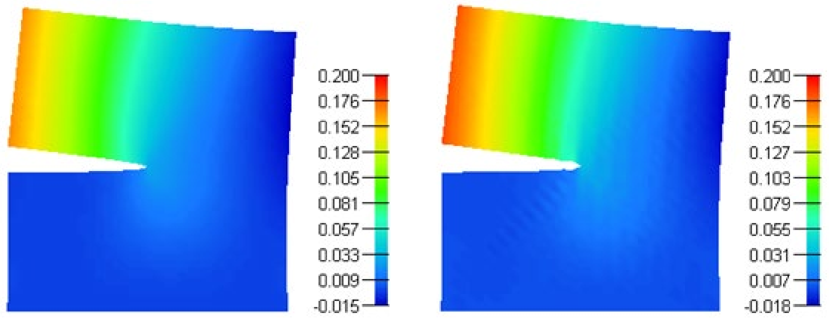


Figure 18: The deformed shapes of the V-notched plate: (a) Abaqus quadrilateral CPE4I and (b) Abaqus NSFEM UEL elements

Table 4: Sharp V-Notch square plate point 'A3' vertical displacement results summary

Reference solution: 0.16986 (m)

Mesh (DOFs)	QUAD CPE4I elements (m)	NSFEM UEL elements (m)
210	0.159277	0.211860
1070	0.167541	0.188262
2210	0.167700	0.182385

5 Conclusions

In this work, Abaqus NSFEM UEL is successfully implemented in Abaqus software for two dimensional linear and nonlinear problems. The following conclusions can be drawn based on the results presented:

1) The proposed Abaqus NSFEM UEL simplifies the preprocessing process as only CST elements are used.
2) The proposed Abaqus NSFEM UEL provides more accurate results compare to the conventional FEM results even when the problem domain is completely discretized with linear triangular elements.
3) This is improvisation in available FEM code. The convergence rate of Abaqus NSFEM UEL results are at par with those obtained from fully integrated Abaqus quadrilateral elements accounting additional complexities like geometric nonlinearity and nearly-incompressible material.
4) The proposed Abaqus NSFEM UEL provides upper bound solution while the conventional FEM provides lower bound solution. Combining these two methods, range bound solution for the problem under consideration can be easily predicted without any major changes.

Acknowledgement: J.J. Yee and C.K. Lee would like to thank the National Research Foundation (NRF) of Korea through Ministry of Education (No. 2016R1A6A1A03012812) for providing financial support for this research undertaking.

References

Abaqus Documentation (2012): *Abaqus standard subroutines. Abaqus User Subroutines Reference Manual*. Dassault Systemes Simulia Corp., Providence, RI, USA.

Altair Hypermesh Documentation (2014): *2D Meshing. Hypermesh 13.0 Manual (Chapter 3)*. Altair Engineering Inc., Troy, Michigan, USA.

Bhatti, A. M. (2006): *Geometric Nonlinearity. Advanced Topics in Finite Element Analysis of Structures: With Mathematica and MATLAB Computations (Chapter 9)*. John Wiley & Sons Ltd., Hoboken, New Jersey, USA.

Bonet, J.; Wood, R. D. (2008): *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, UK.

Chandrupatla, T. R.; Belegundu, A. D. (2016): *Axisymmetric Solids Subjected to Axisymmetric Loading. An Introduction to Finite Element in Engineering (Chapter 7)*. Pearson Indian Education Services Pvt. Ltd, Bengaluru, India.

de Borst, R.; Crisfield, M. A.; Remmers, J. J. C.; Verhoosel, C. V. (2012): *Nonlinear Finite Element Analysis of Solids and Structures*. John Wiley & Sons Ltd., West Sussex, UK.

Fish, J.; Belytschko, T. (2007): *A First Course in Finite Elements*. John Wiley & Sons Ltd., West Sussex, UK.

Kumbhar, P. Y.; Francis, A.; Swaminathan, N.; Annabattula, R. K.; Natarajan, S. (2018): Development of user element routine (UEL) for cell-based smoothed finite element method (CSFEM) in Abaqus. *International Journal of Computational Methods*, vol. 185, pp. 1-28.

Lee, C. K. (2016): *Gradient Smoothing in Finite Elasticity: Near-Incompressibility (Ph.D. Thesis)*. School of Engineering, Cardiff University, UK.

Lee, N.; Bathe, K. (1993): Effects of element distortions on the performance of isoparametric elements. *International Journal for Numerical Methods in Engineering*, vol. 36, no. 20, pp. 3553-3576.

Liu, G. R.; Dai, K. Y.; Nguyen-Thoi, T. (2007): A smoothed finite element method for mechanics problems. *Computational Mechanics*, vol. 39, no. 6, pp. 859-877.

Liu, G. R.; Nguyen-Thoi, T.; Nguyen-Xuan, H.; Lam, K. Y. (2009): A node based smoothed finite element (NS-FEM) for upper bound solution to solid mechanics problems. *Computers and Structures*, vol. 87, no. 1-2, pp. 14-26.

Nguyen-Xuan, H.; Bordas, S.; Nguyen-Dang, H. (2008): Smooth finite element methods: convergence, accuracy and properties. *International Journal for Numerical Methods in Engineering*, vol. 74, no. 2, pp. 175-208.

Reddy, J. N. (2004): *An Introduction to Nonlinear Finite Element Analysis*. Oxford University Press, USA.

Wang, S. (2014): *An Abaqus Implementation of the Cell-Based Smoothed Finite Element Method using Quadrilateral Elements (Ph. D. Thesis)*. University of Cincinnati, USA.

Zeng, W.; Liu, G. R. (2016): Smoothed finite element methods (S-FEM): an overview and recent developments. *Archives of Computational Methods in Engineering*, vol. 25, no. 2, pp. 397-435.