Research Article

Fast Stiffness Matrix Calculation for Nonlinear Finite Element Method

Emir Gülümser,[1] Uğur Güdükbay,[1] and Sinan Filiz[2]

[1] Department of Computer Engineering, Bilkent University, 06800 Ankara, Turkey
[2] Department of Mechanical Engineering, Bilkent University, 06800 Ankara, Turkey

Correspondence should be addressed to Uğur Güdükbay; gudukbay@cs.bilkent.edu.tr

We propose a fast stiffness matrix calculation technique for nonlinear finite element method (FEM). Nonlinear stiffness matrices are constructed using Green-Lagrange strains, which are derived from infinitesimal strains by adding the nonlinear terms discarded from small deformations. We implemented a linear and a nonlinear finite element method with the same material properties to examine the differences between them. We verified our nonlinear formulation with different applications and achieved considerable speedups in solving the system of equations using our nonlinear FEM compared to a state-of-the-art nonlinear FEM.

1. Introduction

Mesh deformations have widespread usage areas, such as computer games, computer animations, fluid flow, heat transfer, surgical simulations, cloth simulations, and crash test simulations. The major goal in mesh deformations is to establish a good balance between the accuracy of the simulation and the computational cost; achieving this balance depends on the application. The speed of the simulation is far more important than the accuracy in computer games. The simulation needs to be real-time in order to be used in games so free-form deformation or fast linear FEM solvers can be used. However, high computation cost gives much more accurate results when we are working with life-critical applications such as car crash tests, surgical simulators, and concrete analysis of buildings; even linear FEM solvers are not adequate enough for these types of applications in terms of the accuracy.

For realistic and highly accurate deformations, one can use the finite element method (FEM), a numerical technique to find approximate solutions to engineering and mathematical physics problems. FEM could be used to solve problems in areas such as structural analysis, heat transfer, fluid flow, mass transport, and electromagnetics [1, 2].

We propose a fast stiffness matrix calculation technique for nonlinear FEM. We derive nonlinear stiffness matrices using Green-Lagrange strains, themselves derived from infinitesimal strains by adding the nonlinear terms discarded from infinitesimal strain theory.

We mainly focus on the construction of the stiffness matrices because change in material parameters and change in boundary conditions can be directly represented and applied without choosing a proper FEM [3]. Joldes et al. [4] and Taylor et al. [5] achieve real-time computations of soft tissue deformations for nonlinear FEM using GPUs; however, they do not describe how they compute stiffness matrices; thus, we cannot implement their methods and compare them with our proposed method. Cerrolaza and Osorio describe a simple and efficient method to reduce the integration time of nonlinear FEM for dynamic problems using hexahedral 8-noded finite elements [6]. We compare our stiffness matrix calculations with Pedersen's method [3] to measure performance and verify correctness. We achieve a 142% speedup in calculating the stiffness matrices and a 15%

speedup in solving the whole system on average, compared to Pedersen's method.

2. The Nonlinear FEM with Green-Lagrange Strains

We use tetrahedral elements for modeling meshes in the experiments. Overall, there are 12 unknown nodal displacements in a tetrahedral element. They are given by [2]

$$\{d\} = \{u(x, y, z)\} = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \vdots \\ u_4 \\ v_4 \\ w_4 \end{Bmatrix}. \tag{1}$$

In global coordinates, we represent displacements by linear function by

$$u^e(x, y, z) = c_1 + c_2 x + c_3 y + c_4 z. \tag{2}$$

For all 4 vertices, (2) is extended as

$$\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}. \tag{3}$$

Constants c_n can be found as

$$c_n = v^{-1} u_n, \tag{4}$$

where v^{-1} is given by

$$v^{-1} = \frac{1}{\det(v)} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix}. \tag{5}$$

$\det(v)$ is $6V$, where V is the volume of the tetrahedron. If we substitute (4) into (2), we obtain

$$u^e(x, y, z) = \frac{1}{6V^e} \begin{bmatrix} 1 & x & y & z \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}. \tag{6}$$

$\alpha, \beta, \gamma, \delta$, and the volume V are calculated by

$$\alpha_1 = \begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}, \qquad \alpha_2 = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix},$$

$$\alpha_3 = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_4 & y_4 & z_4 \end{vmatrix}, \qquad \alpha_4 = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix},$$

$$\beta_1 = -\begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix}, \qquad \beta_2 = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix},$$

$$\beta_3 = -\begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_4 & z_4 \end{vmatrix}, \qquad \beta_4 = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix},$$

$$\gamma_1 = \begin{vmatrix} 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{vmatrix}, \qquad \gamma_2 = -\begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{vmatrix},$$

$$\gamma_3 = \begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_4 & z_4 \end{vmatrix}, \qquad \gamma_4 = -\begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \end{vmatrix},$$

$$\delta_1 = -\begin{vmatrix} 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{vmatrix}, \qquad \delta_2 = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{vmatrix},$$

$$\delta_3 = -\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_4 & y_4 \end{vmatrix}, \qquad \delta_4 = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix},$$

$$6V = \begin{vmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \\ 1 & x_l & y_l & z_l \end{vmatrix}. \tag{7}$$

Because of the differentials in strain calculation, α is not used in the following stages. If we expand (6), we obtain

$$u^e(x, y, z)$$
$$= \frac{1}{6V^e} \begin{bmatrix} \alpha_1 + \beta_1 x + \gamma_1 y + \delta_1 z \\ \alpha_2 + \beta_2 x + \gamma_2 y + \delta_2 z \\ \alpha_3 + \beta_3 x + \gamma_3 y + \delta_3 z \\ \alpha_4 + \beta_4 x + \gamma_4 y + \delta_4 z \end{bmatrix} \tag{8}$$
$$\times \begin{bmatrix} u(x, y, z)_1 & u(x, y, z)_2 & u(x, y, z)_3 & u(x, y, z)_4 \end{bmatrix}.$$

For tetrahedral elements, to express displacements in simpler form, shape functions are introduced $(\psi_1, \psi_2, \psi_3, \psi_4)$. They are given by

$$\psi_1 = \frac{1}{6V} (\alpha_1 + \beta_1 x + \gamma_1 y + \delta_1 z) u(x, y, z)_1,$$

$$\psi_2 = \frac{1}{6V} (\alpha_2 + \beta_2 x + \gamma_2 y + \delta_2 z) u(x, y, z)_2,$$

$$\psi_3 = \frac{1}{6V} (\alpha_3 + \beta_3 x + \gamma_3 y + \delta_3 z) u(x, y, z)_3, \tag{9}$$

$$\psi_4 = \frac{1}{6V} (\alpha_4 + \beta_4 x + \gamma_4 y + \delta_4 z) u(x, y, z)_4.$$

In our method, nonlinear stiffness matrices are derived using Green-Lagrange strains (large deformations), which
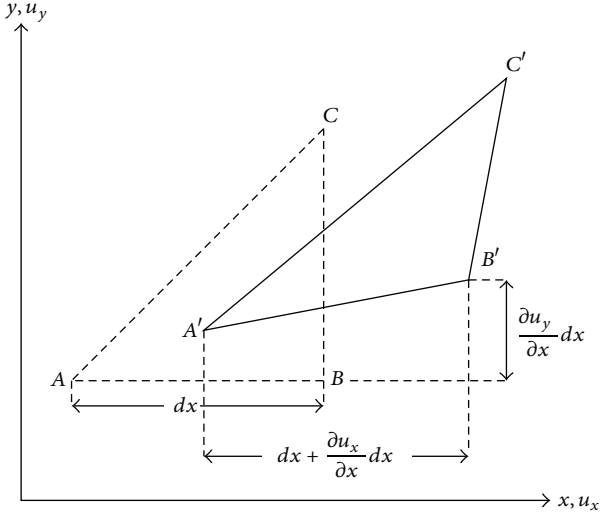
FIGURE 1: 2D element before and after deformation.

themselves are derived directly from infinitesimal strains (small deformations), by adding the nonlinear terms discarded in infinitesimal strain theory. The proposed nonlinear FEM uses the linear FEM framework but it does not require the explicit use of weight functions and differential equations. Hence, numerical integration is not needed for the solution of the proposed nonlinear FEM. Instead of using weight functions and integrals, we use displacement gradients and strains to make the elemental stiffness matrices space-independent in order to discard the integral. We extend the linear FEM to the nonlinear FEM by extending the linear strains to the Green-Lagrange strains.

We constructed our linear FEM by extending Logan's 2D linear FEM to 3D [2]. To understand Green-Lagrange strains, we must first see how they differ from the infinitesimal strains used to calculate the global stiffness matrices in a linear FEM. Figure 1 shows a 2D element before and after deformation, where the element edge AB with initial length dx becomes $A'B'$. The engineering normal strain is calculated as the change in the length of the line

$$\varepsilon_x = \frac{\left|A'B'\right| - |AB|}{|AB|}. \tag{10}$$

The final length of the elemental edge can be calculated using

$$\left|A'B'\right|^2 = \left(dx + \frac{\partial u_x}{\partial x}dx\right)^2 + \left(\frac{\partial u_y}{\partial x}dx\right)^2,$$

$$\left|A'B'\right|^2 = dx^2\left[1 + 2\left(\frac{\partial u_x}{\partial x}\right) + \left(\frac{\partial u_x}{\partial x}\right)^2 + \left(\frac{\partial u_y}{\partial x}\right)^2\right]. \tag{11}$$

By neglecting the higher-order terms in (11), 2D infinitesimal strains are defined by

$$\varepsilon_{xx} = \frac{\partial u_x}{\partial x}, \qquad \varepsilon_{yy} = \frac{\partial u_y}{\partial y}, \qquad \varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right). \tag{12}$$

By the definition, the nonlinear FEM differs from the linear FEM because of the nonlinearity that arises from the higher-order term neglected in calculation of strains. The strain vector used in the linear FEM relies on the assumption that the displacements at the x-axis, y-axis, and z-axis are very small. The initial and final positions of a given particle are practically the same; thus, the higher-order terms are neglected [7]. When the displacements are large, however, this is no longer the case and one must distinguish between the initial and final coordinates of the particles; thus the higher-order terms are added into the strain equations. By adding these high-order terms, 3D strains are defined as [8]

$$\eta_{xx} = \frac{\partial u_x}{\partial x} + \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial x}\right) + \left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial x}\right) + \left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial x}\right)\right],$$

$$\eta_{yy} = \frac{\partial u_y}{\partial y} + \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial y}\right) + \left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial y}\right) + \left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial y}\right)\right],$$

$$\eta_{zz} = \frac{\partial u_z}{\partial z} + \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial z}\right) + \left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial z}\right) + \left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial z}\right)\right],$$

$$\eta_{xy} = \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right)$$
$$+ \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial y}\right) + \left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial y}\right) + \left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial y}\right)\right],$$

$$\eta_{zx} = \frac{1}{2}\left(\frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}\right)$$
$$+ \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial x}\right) + \left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial x}\right) + \left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial x}\right)\right],$$

$$\eta_{yz} = \frac{1}{2}\left(\frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y}\right)$$
$$+ \frac{1}{2}\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial z}\right) + \left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial z}\right) + \left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial z}\right)\right], \tag{13}$$

which leads to

$$\{n\} = \begin{Bmatrix} \eta_{xx} \\ \eta_{yy} \\ \eta_{zz} \\ 2\left(\eta_{xy} + \eta_{yx}\right) \\ 2\left(\eta_{xz} + \eta_{zx}\right) \\ 2\left(\eta_{yz} + \eta_{zy}\right) \end{Bmatrix} = \begin{Bmatrix} \eta_{xx} \\ \eta_{yy} \\ \eta_{zz} \\ 2\eta_{xy} \\ 2\eta_{zx} \\ 2\eta_{yz} \end{Bmatrix}. \tag{14}$$

The Green-Lagrange strain tensor is represented in matrix notation as

$$\{\eta\} = \left[B_L^T\right]\{d\} + \frac{1}{2}\{d\}^T\left[B_{\mathrm{NL}}\right]\{d\}, \tag{15}$$

where $\{d\}$ is the nodal displacement, $[B_L]$ is the linear, and $[B_{\mathrm{NL}}]$ is the nonlinear part of the $[B_0]$ matrix [3]. For a specific

element, $[B_L]$ and $[B_{NL}]$ are constant, as with the $[B]$ matrix in the linear FEM. With the variation of $\{d\}$ [9], (15) becomes

$$\{\overline{\eta}\} = \left[B_L^T\right]\{d\} + \{d\}^T\left[B_{NL}\right]\{d\}. \tag{16}$$

Gathering the strain components together, we can rewrite (15) and (16) as

$$\{\eta\} = \left([B_L] + \frac{1}{2}\{d^T\}[B_{NL}]\right)\{d\} = [B_0]\{d\}, \tag{17}$$

$$\{\overline{\eta}\} = \left([B_L] + \{d^T\}[B_{NL}]\right)\{d\} = \left[\overline{B_0}\right]\{d\}.$$

The linear part of the $[B_0]$ matrix ($[B_L]$) is the same as the $[B]$ matrix in the linear FEM. Calculating $[B_0]$ becomes more complex with the introduction of the nonlinear terms. After finding the nonlinear strains, these equations are combined with the shape functions to find matrix $[B_0]$:

$$\{\overline{\eta}\} = \left[\overline{B_0}\right]\{d\}. \tag{18}$$

The most frequently used terms, which are the nine displacement gradients for calculating the nonlinear strains, are $\partial u_x/\partial x$, $\partial u_x/\partial y$, $\partial u_x/\partial z$, $\partial u_y/\partial x$, $\partial u_y/\partial y$, $\partial u_y/\partial x$, $\partial u_z/\partial x$, $\partial u_z/\partial y$, and $\partial u_z/\partial z$. Using (8) for displacements, we can construct the displacement gradients using the partial derivatives of the shape functions. They are represented by

$$u_{xx} = \frac{1}{6V}\left(\beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 + \beta_4 u_4\right),$$

$$u_{yx} = \frac{1}{6V}\left(\beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 + \beta_4 v_4\right),$$

$$u_{zx} = \frac{1}{6V}\left(\beta_1 w_1 + \beta_2 w_2 + \beta_3 w_3 + \beta_4 w_4\right),$$

$$u_{xy} = \frac{1}{6V}\left(\gamma_1 u_1 + \gamma_2 u_2 + \gamma_3 u_3 + \gamma_4 u_4\right),$$

$$u_{yy} = \frac{1}{6V}\left(\gamma_1 v_1 + \gamma_2 v_2 + \gamma_3 v_3 + \gamma_4 v_4\right), \tag{19}$$

$$u_{zy} = \frac{1}{6V}\left(\gamma_1 w_1 + \gamma_2 w_2 + \gamma_3 w_3 + \gamma_4 w_4\right),$$

$$u_{xz} = \frac{1}{6V}\left(\delta_1 u_1 + \delta_2 u_2 + \delta_3 u_3 + \delta_4 u_4\right),$$

$$u_{yz} = \frac{1}{6V}\left(\delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3 + \delta_4 v_4\right),$$

$$u_{zz} = \frac{1}{6V}\left(\delta_1 w_1 + \delta_2 w_2 + \delta_3 w_3 + \delta_4 w_4\right),$$

where u_{xx} represents $\partial u_x/\partial x$.

We can evaluate the partial derivatives of the shape functions as follows (for the 1st node of $[B_{NL}]$):

$$\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial x}\right) + \left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial x}\right) + \left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial x}\right)\right]$$

$$= \frac{1}{6V}\left(\beta_1\left(u_{xx} + u_{yx} + u_{zx}\right)\right),$$

$$\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial y}\right) + \left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial y}\right) + \left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial y}\right)\right]$$

$$= \frac{1}{6V}\left(\gamma_1\left(u_{xy} + u_{yy} + u_{zy}\right)\right),$$

$$\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial z}\right) + \left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial z}\right) + \left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial z}\right)\right]$$

$$= \frac{1}{6V}\left(\delta_1\left(u_{xz} + u_{yz} + u_{zz}\right)\right),$$

$$\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial y}\right) + \left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial y}\right) + \left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial y}\right)\right]$$

$$= \frac{1}{6V}\left(\gamma_1\left(u_{xx} + u_{yx} + u_{zx}\right)\right)$$

$$+ \frac{1}{6V}\left(\beta_1\left(u_{xy} + u_{yy} + u_{zy}\right)\right),$$

$$\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial x}\right) + \left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial x}\right) + \left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial x}\right)\right]$$

$$= \frac{1}{6V}\left(\delta_1\left(u_{xx} + u_{yx} + u_{zx}\right)\right)$$

$$+ \frac{1}{6V}\left(\beta_1\left(u_{xz} + u_{yz} + u_{zz}\right)\right),$$

$$\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial z}\right) + \left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial z}\right) + \left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial z}\right)\right]$$

$$= \frac{1}{6V}\left(\gamma_1\left(u_{xz} + u_{yz} + u_{zz}\right)\right)$$

$$+ \frac{1}{6V}\left(\delta_1\left(u_{xy} + u_{yy} + u_{zy}\right)\right). \tag{20}$$

Using (17) and (20), we obtain $\left[\overline{B_0}\right]$ for the 1st node (21). Similarly, using (17) and (20), we obtain $[B_0]$ for the 1st node (22).

Consider

$$
\left[\overline{B_{01}}\right] =
\begin{bmatrix}
\beta_1 + \beta_1\left(u_{xx}\right) & \beta_1\left(u_{yx}\right) & \beta_1\left(u_{zx}\right) \\
\gamma_1\left(u_{xy}\right) & \gamma_1 + \gamma_1\left(u_{yy}\right) & \gamma_1\left(u_{zy}\right) \\
\delta_1\left(u_{xz}\right) & \delta_1\left(u_{yz}\right) & \delta_1 + \delta_1\left(u_{zz}\right) \\
\gamma_1 + \gamma_1\left(u_{xx}\right) + \beta_1\left(u_{xy}\right) & \gamma_1\left(u_{yx}\right) + \beta_1 + \beta_1\left(u_{yy}\right) & \gamma_1\left(u_{zx}\right) + \beta_1\left(u_{zy}\right) \\
\delta_1 + \delta_1\left(u_{xx}\right) + \beta_1\left(u_{xz}\right) & \delta_1\left(u_{yx}\right) + \beta_1\left(u_{yz}\right) & \delta_1\left(u_{zx}\right) + \beta_1 + \beta_1\left(u_{zz}\right) \\
\gamma_1\left(u_{xz}\right) + \delta_1\left(u_{xy}\right) & \gamma_1 + \gamma_1\left(u_{yz}\right) + \delta_1\left(u_{yy}\right) & \gamma_1\left(u_{zz}\right) + \delta_1 + \delta_1\left(u_{zy}\right)
\end{bmatrix}
\begin{Bmatrix} u_1 \\ v_1 \\ w_1 \end{Bmatrix},
\tag{21}
$$

$$
\left[B_{0_1}\right] =
\begin{bmatrix}
\beta_1 + \dfrac{1}{2}\beta_1\left(u_{xx}\right) & \dfrac{1}{2}\beta_1\left(u_{yx}\right) & \dfrac{1}{2}\beta_1\left(u_{zx}\right) \\
\dfrac{1}{2}\gamma_1\left(u_{xy}\right) & \gamma_1 + \dfrac{1}{2}\gamma_1\left(u_{yy}\right) & \dfrac{1}{2}\gamma_1\left(u_{zy}\right) \\
\dfrac{1}{2}\delta_1\left(u_{xz}\right) & \dfrac{1}{2}\delta_1\left(u_{yz}\right) & \delta_1 + \dfrac{1}{2}\delta_1\left(u_{zz}\right) \\
\gamma_1 + \dfrac{1}{2}\gamma_1\left(u_{xx}\right) + \dfrac{1}{2}\beta_1\left(u_{xy}\right) & \dfrac{1}{2}\gamma_1\left(u_{yx}\right) + \beta_1 + \dfrac{1}{2}\beta_1\left(u_{yy}\right) & \dfrac{1}{2}\gamma_1\left(u_{zx}\right) + \dfrac{1}{2}\beta_1\left(u_{zy}\right) \\
\delta_1 + \dfrac{1}{2}\delta_1\left(u_{xx}\right) + \dfrac{1}{2}\beta_1\left(u_{xz}\right) & \dfrac{1}{2}\delta_1\left(u_{yx}\right) + \dfrac{1}{2}\beta_1\left(u_{yz}\right) & \dfrac{1}{2}\delta_1\left(u_{zx}\right) + \beta_1 + \dfrac{1}{2}\beta_1\left(u_{zz}\right) \\
\dfrac{1}{2}\gamma_1\left(u_{xz}\right) + \dfrac{1}{2}\delta_1\left(u_{xy}\right) & \gamma_1 + \dfrac{1}{2}\gamma_1\left(u_{yz}\right) + \dfrac{1}{2}\delta_1\left(u_{yy}\right) & \dfrac{1}{2}\gamma_1\left(u_{zz}\right) + \delta_1 + \dfrac{1}{2}\delta_1\left(u_{zy}\right)
\end{bmatrix}
\begin{Bmatrix} u_1 \\ v_1 \\ w_1 \end{Bmatrix}.
\tag{22}
$$

The FEM is derived from conservation of the potential energy, which is defined by

$$
\pi = \mathbf{E}_{\text{strain}} + W,
\tag{23}
$$

where $\mathbf{E}_{\text{strain}}$ is the strain energy of the linear element and W is the work potential. They are given by

$$
\mathbf{E}_{\text{strain}} = \frac{1}{2}\int_{\Omega^e} \varepsilon^T \sigma\, dx, \qquad W = f^e d^T,
\tag{24}
$$

where the engineering strain vector $\{\varepsilon\}$ is

$$
\{\varepsilon\} = [B]\{d\}.
\tag{25}
$$

From (24), the engineering stress vector τ is related to the strain vector by

$$
\tau = [E]\{\overline{\eta}\} = [E]\left[\overline{B_0}\right]\{d\}.
\tag{26}
$$

The secant relations are described by the matrix $[E]$. We substitute (15) and (26) into (24), obtaining the element stiffness matrix

$$
[k(u)] = \iiint \{d\}^T [B_0]^T [E]\left[\overline{B_0}\right]\{d\}\, dx\, dy\, dz.
\tag{27}
$$

We can discard the integrals as we did for the linear FEM. $[B_0]$, $[E]$, and $[\overline{B_0}]$ are constant for the four-node tetrahedral element, so (27) is rewritten as

$$
[k(u)] = \{d\}^T [B_0]^T [E]\left[\overline{B_0}\right] V.
\tag{28}
$$

The secant stiffness matrix which is $[k_s(d)^T] = [B_0]^T [E][\overline{B_0}]$ is nonsymmetric because of the fact that $[B_0]^T \neq [\overline{B_0}]$.

Introducing nodal forces, we obtain

$$
\{f\} =
\begin{Bmatrix}
f_{1x} \\ f_{1y} \\ f_{1z} \\ \vdots \\ f_{4x} \\ f_{4y} \\ f_{4z}
\end{Bmatrix}
\{d\}^T.
\tag{29}
$$

With the equilibrium equation and cancelling $\{d\}^T$, the whole system for one element reduces to

$$
k_s(d)^e \{d\}^e = f^e.
\tag{30}
$$

By substituting $\{d\}$ with u, we obtain

$$
k_s(u)^e u^e = f^e.
\tag{31}
$$

Finally, only nonlinear displacement functions remain, which are solved with Newton-Raphson to find the unknown displacements u [10].

Element residuals are necessary for the iterative Newton-Raphson method. The element residual is a 12×1 vector for a specific element. The residual for a specific element is defined as

$$
r^e = k_s(u)^e - f^e.
\tag{32}
$$

Having determined r^e, we can now express (32) in expanded vector form as

$$\begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{12} \end{Bmatrix} = \begin{bmatrix} k_s(u)_{(1,1)} + k_s(u)_{(1,2)} + k_s(u)_{(1,3)} + \cdots + k_s(u)_{(1,12)} \\ k_s(u)_{(2,1)} + k_s(u)_{(2,2)} + k_s(u)_{(2,3)} + \cdots + k_s(u)_{(2,12)} \\ k_s(u)_{(3,1)} + k_s(u)_{(3,2)} + k_s(u)_{(3,3)} + \cdots + k_s(u)_{(3,12)} \\ \vdots \\ k_s(u)_{(12,1)} + k_s(u)_{(12,2)} + k_s(u)_{(12,3)} + \cdots + k_s(u)_{(12,12)} \end{bmatrix} - \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{12} \end{Bmatrix}. \tag{33}$$

The tangent stiffness matrix $[K]_T^e(r'^e)$ is also necessary for the iterative Newton-Raphson method. The tangent stiffness matrix is also 12×12 matrix, like the elemental stiffness matrix. However, the tangent stiffness matrix depends on residuals, unlike the elemental stiffness matrix. Elemental stiffness matrices are used to construct residuals and the derivatives of the residuals are used to construct the elemental tangent stiffness matrices. We can express the elemental tangent stiffness matrix for a specific element as

$$r'^e = [K]_T^e = \begin{bmatrix} \dfrac{\partial}{\partial u_1} r_1 & \dfrac{\partial}{\partial v_1} r_1 & \dfrac{\partial}{\partial w_1} r_1 & \cdots & \dfrac{\partial}{\partial u_4} r_1 & \dfrac{\partial}{\partial v_4} r_1 & \dfrac{\partial}{\partial w_4} r_1 \\[2mm] \dfrac{\partial}{\partial u_1} r_2 & \dfrac{\partial}{\partial v_1} r_2 & \dfrac{\partial}{\partial w_1} r_2 & \cdots & \dfrac{\partial}{\partial u_4} r_2 & \dfrac{\partial}{\partial v_4} r_2 & \dfrac{\partial}{\partial w_4} r_2 \\[2mm] \dfrac{\partial}{\partial u_1} r_3 & \dfrac{\partial}{\partial v_1} r_3 & \dfrac{\partial}{\partial w_1} r_3 & \cdots & \dfrac{\partial}{\partial u_4} r_3 & \dfrac{\partial}{\partial v_4} r_3 & \dfrac{\partial}{\partial w_4} r_3 \\[2mm] \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\[2mm] \dfrac{\partial}{\partial u_1} r_{12} & \dfrac{\partial}{\partial v_1} r_{12} & \dfrac{\partial}{\partial w_1} r_{12} & \cdots & \dfrac{\partial}{\partial u_4} r_{12} & \dfrac{\partial}{\partial v_4} r_{12} & \dfrac{\partial}{\partial w_4} r_{12} \end{bmatrix}. \tag{34}$$

Newton-Raphson method is a fast and popular numerical method for solving nonlinear equations [10], as compared to the other methods, such as direct iteration. In principle, the method works by applying the following two steps (cf. Algorithm 1): (i) check if the equilibrium is reached within the desired accuracy; (ii) if not, make a suitable adjustment to the state of the deformation [11]. An initial guess for displacements is needed to start the iterations. The displacements are updated according to

$$x_{k+1} = x_k - \frac{f_{x_k}}{f'_{x_k}}. \tag{35}$$

In the proposed nonlinear FEM, u is the vector that keeps the information of the nodal displacements. Instead of making only one assumption, we make whole u vector initial guess in order to start the iteration.

Consider

$$u_1 = u_0 - \frac{r_{u_0}}{r'_{u_0}}, \tag{36}$$

where r is residual of the global stiffness matrix $[K]$ calculated in (33) and r' is the tangent stiffness matrix calculated in (34).

At every step, the vector r and the matrix r' are updated for every element with the new u_i values. Then, r and r' are assembled as we did with for the global stiffness matrix K and the global force vector F in linear FEM. Boundary conditions are applied to the global r vector and the global r' matrix. Using the global r vector and the global r' matrix, we have

$$r'(u_i) p = -r(u_i), \quad p = -\left(r'(u_i)\right)^{-1} r(u_i). \tag{37}$$

u_i is updated with the solution of (37). Consider

$$u_{i+1} = u_i + p. \tag{38}$$

Then, we check if the equilibrium is reached within the desired accuracy defined by δ as

$$|r(u_i)| \le \delta. \tag{39}$$

After the desired accuracy is reached, the unknown nodal displacements are found.
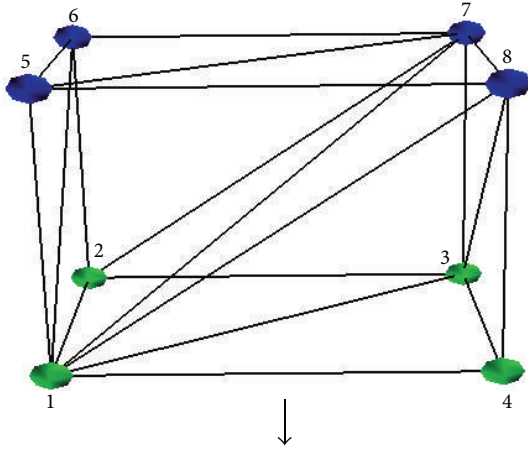
3. Experimental Results

The proposed nonlinear FEM and Pedersen's nonlinear FEM were implemented using MATLAB programming language.

```
Make initial guess f(x)
while |f(x)| ≤ δ do
    Compute p = − f(x)/f'(x)
    Update x = x + p
    Calculate f(x)
end while
```

ALGORITHM 1: Newton-Raphson method.

FIGURE 2: A $10 \times 10 \times 10$ m cube mesh with eight nodes and six tetrahedra is constrained at the blue nodes and pulled downwards from the green nodes.

The visualizer was implemented with C++ language and connected to the solver using the MATLAB engine [12], which allows users to call the MATLAB solver from C/C++ or Fortran programs. The simulation results, interaction with the 3D model, and the 3D models themselves were visualized using OpenGL, and the nose experiment was visualized using 3ds Max [13]. To speed up the nonlinear FEM, we used MAPLE's symbolic solver [14], which is integrated into MATLAB. We conducted all the experiments on a desktop computer with a Core i7 3930 K processor overclocked at 4.2 GHz with 32 GB of RAM. We used linear material properties for the models in the experiments. We used 1 GPa for Young's modulus (ϵ) because polypropylene has Young's modulus between 1.5 and 2 GPa and polyethylene HDPE has Young's modulus 0.8 GPa, which shows plastic properties and they are close to 1 GPa. Because most steels and plastic materials undergo plastic deformation near the value of 0.3, we used 0.25 for Poisson's ratio (ν). Our simulation is static so we used a single load step in all experiments. Multiple load steps are used when the load forces are time-dependent or the simulation is dynamic [15].

We conducted four experiments, each having different number of elements to observe the speedup for both stiffness matrix calculation and for the solution of the system. As expected, the proposed method and Pedersen's method produced same amount of nodal displacements in all experiments.

TABLE 1: The displacements (in m) at nodes 1, 2, 3, and 4 using the linear FEM for the first experiment. The displacements of nodes 5 to 8 for all axes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
1	0.027234	0.011064	−0.289965
2	0.004306	−0.109719	−0.440739
3	−0.066065	−0.056547	−0.343519
4	−0.107536	0.070143	−0.514524

TABLE 2: The displacements (in m) at nodes 1, 2, 3, and 4 using the nonlinear FEM for the first experiment. The displacements of nodes 5 to 8 for all axes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
1	0.029911	0.012665	−0.278365
2	0.008606	−0.103350	−0.415594
3	−0.058835	−0.051901	−0.324126
4	−0.098945	0.068928	−0.478495

TABLE 3: The displacements (in m) at green nodes using the linear FEM for the second experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
0	−3.717	4.208	−0.0394
1	4.738	4.208	−0.04947
2	4.737	−4.245	0.03777
3	−3.716	−4.246	0.04902
20	3.01	−3.547	−0.05429
21	−4.117	−3.548	−0.06143
22	−4.117	3.581	0.04348
23	3.01	3.581	0.05155

TABLE 4: The displacements (in m) at green nodes using the nonlinear FEM for the second experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
0	−2.083	4.102	0.3798
1	4.72	2.298	0.3588
2	2.913	−4.501	0.4586
3	−3.884	−2.699	0.4809
20	3.28	−2.561	−0.424
21	−2.842	−3.931	−0.4032
22	−4.217	2.194	−0.3018
23	1.911	3.565	−0.3212

The first experiment was conducted for a cube mesh with eight nodes and six tetrahedral elements. Figure 2 shows that the cube is constrained at the upper four nodes and pulled downwards with a small amount of force (one unit force for each of the upper four nodes). This experiment was conducted with a small mesh in order to carefully examine the nodal displacements and strains for each element. Figure 3 shows the initial and final positions of the nodes for the linear and nonlinear FEMs, respectively. As seen in Figure 3, the

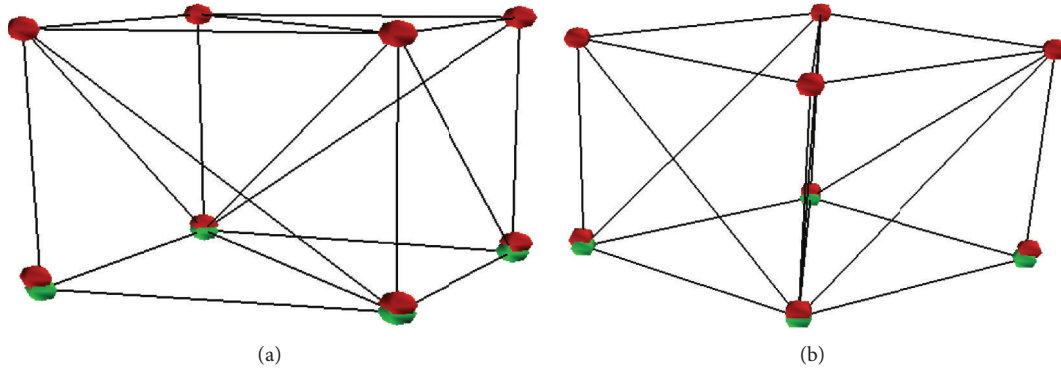(a)   (b)

FIGURE 3: (a) The initial and final positions of the nodes for the linear FEM. (b) The initial and final positions of the nodes for the nonlinear FEM.
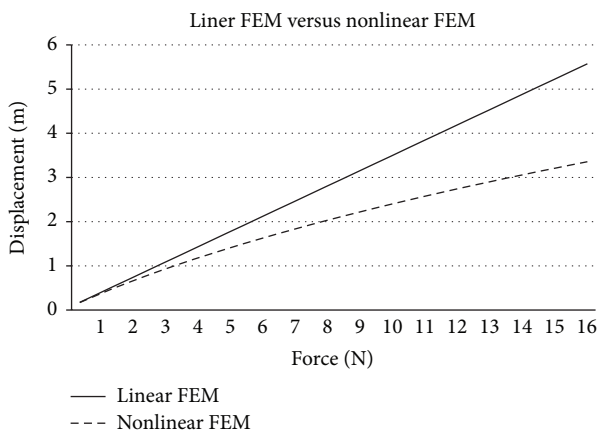

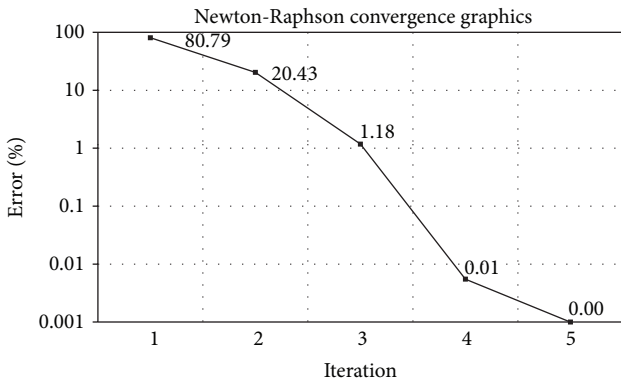FIGURE 4: Force displacements (in m) at node 4 for the linear and nonlinear FEMs.

FIGURE 5: Newton-Raphson convergence graphics for the nonlinear FEM. The graph is plotted using the logarithmic scale.

TABLE 5: The displacements (in m) at green nodes using the linear FEM for the third experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
5	5.004	230.5	7.241
6	−0.4613	239.9	0.01444
9	−2.3	231.8	6.271
10	−4.602	237.5	−0.1437
41	0.5991	234.5	−0.17

TABLE 6: The displacements (in m) at green nodes using the nonlinear FEM for the third experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
5	6.439	69.34	5.014
6	−0.2123	79.96	1.458
9	−3.372	44.71	−4.788
10	0.5483	77.37	0.06587
41	1.196	82.84	0.1512

TABLE 7: The displacements (in m) at green nodes using the linear FEM for the fourth experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
271	1.086	−0.5297	11.88

TABLE 8: The displacements (in m) at green nodes using the nonlinear FEM for the fourth experiment. The displacements at blue nodes are zero.

Node	Displacement-x	Displacement-y	Displacement-z
271	0.6538	−0.1851	4.22

linear and nonlinear methods produce similar displacements when the force magnitude is small. Tables 1 and 2 show the displacements at force applied nodes (first, second, third, and fourth) using the linear and nonlinear FEMs, respectively. Figure 4 shows that displacement increases linearly with force magnitude. However, as expected, the nonlinear

FEM behaves quadratically due to the nonlinear strain definitions. Figure 5 depicts the convergence of the Newton-Raphson method for the nonlinear FEM.

The second experiment was conducted on a beam with 90 nodes and 216 tetrahedral elements. Figures 6(a) and 6(b) show that the beam is constrained at the blue nodes and
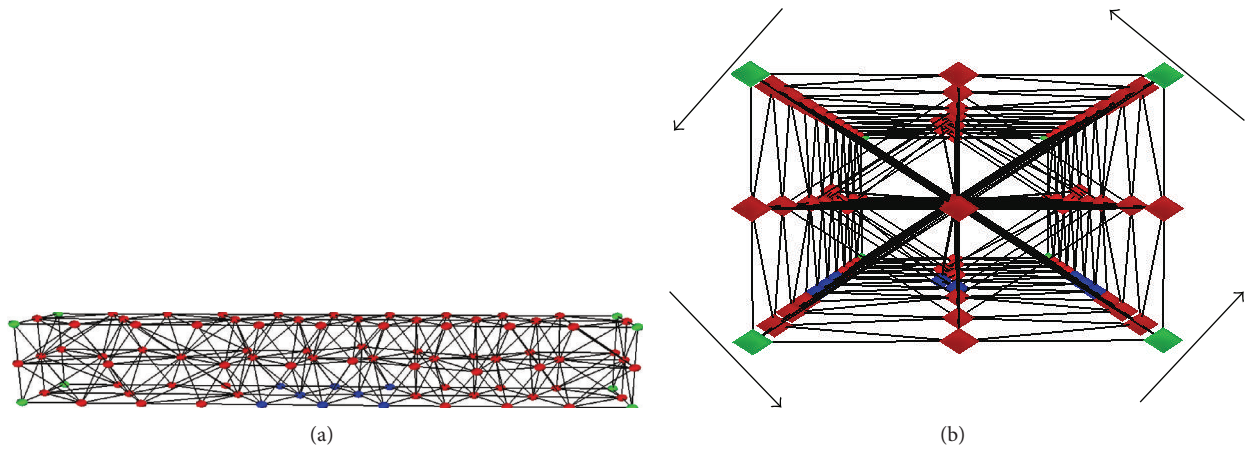
(a)

(b)

FIGURE 6: The beam mesh is constrained at the blue nodes and twisted at the green nodes. (a) Front view; (b) side view, which shows the force directions applied on each green node.
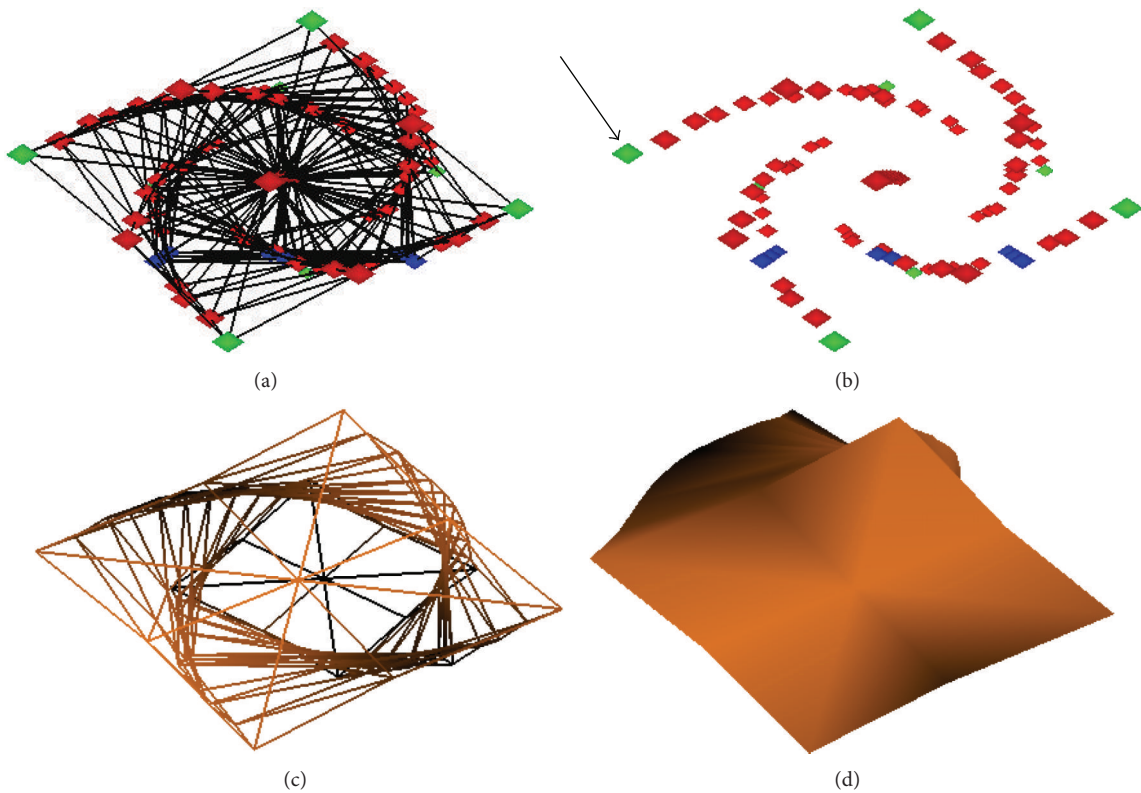

(a)

(b)

(c)

(d)

FIGURE 7: Nonlinear FEM results for the both the proposed and Pedersen's methods ((a) wireframe tetrahedra and nodes; (b) nodes only; (c) wireframe surface mesh; and (d) shaded mesh).
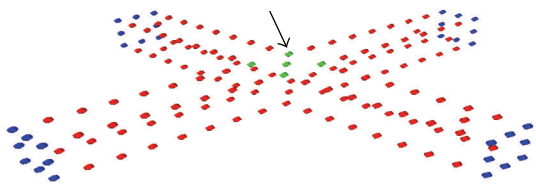

FIGURE 8: The cross mesh is constrained at the blue nodes and pushed towards the green nodes.

twisted at both ends. Figure 7 shows the final shape of the beam mesh for both the proposed and Pedersen's methods. Tables 3 and 4 show the displacements at force applied nodes (green nodes) for the second experiment using the linear and nonlinear FEMs, respectively.

The third experiment was conducted with a cross mesh of 159 nodes and 244 tetrahedral elements. We aimed to observe if there is a root jump occurring when solving the system for a high amount of force (50 N units), and its effect
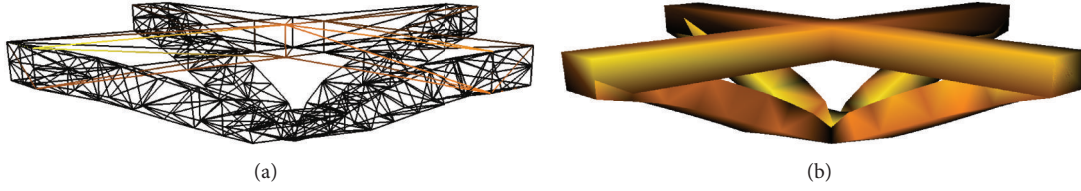
(a)

(b)

FIGURE 9: Nonlinear FEM results for the both proposed and Pedersen's methods: (a) initial and final wireframe meshes are overlaid; (b) initial and final shaded meshes are overlaid.
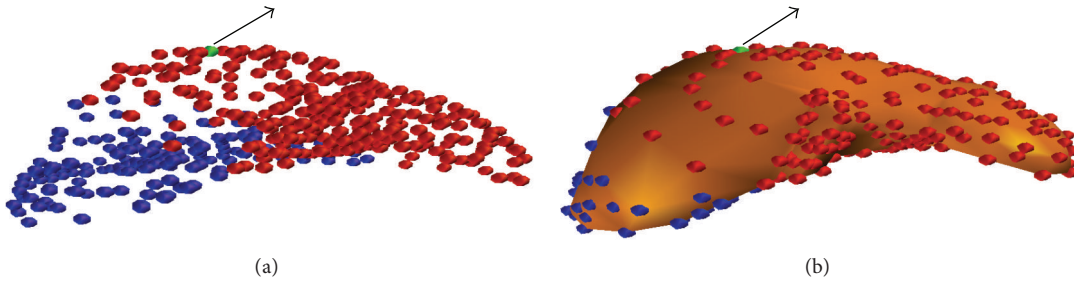

(a)

(b)

FIGURE 10: The liver mesh is constrained at the blue nodes and pulled from the green node ((a) initial nodes; (b) initial shaded mesh and nodes).

TABLE 9: Computation times (in seconds) of stiffness matrices (Pedersen: Pedersen's nonlinear FEM; Proposed: proposed nonlinear FEM; and Speed-up: relative performance comparison of the stiffness matrix calculation of the proposed nonlinear FEM method with Pedersen's nonlinear FEM method using single thread).

Exp.	Elements	Pedersen	Proposed	Speed-up
1st	6	0.7322	0.3308	221.3422
2nd	216	28.1624	11.1542	252.4825
3rd	224	30.1094	12.2725	245.3404
4th	1580	239.8753	96.7840	247.8460

TABLE 10: Computation times (s) of system solutions (Pedersen: Pedersen's nonlinear FEM; Proposed: proposed nonlinear FEM; and Speed-up: relative performance comparison of the proposed nonlinear FEM method with Pedersen's nonlinear FEM method using single thread).

Exp.	Elements	Pedersen	Proposed	Speed-up
1st	6	3.0144	2.4427	123.4044
2nd	216	192.5288	159.6241	120.6139
3rd	224	586.2708	612.5911	95.7034
4th	1580	2840.7558	2401.0994	118.3106

TABLE 11: Newton-Raphson iteration count to reach desired accuracy (Pedersen: Pedersen's nonlinear FEM; Proposed: proposed nonlinear FEM).

Exp.	Elements	Pedersen	Proposed
1st	6	5	5
2nd	216	7	7
3rd	224	26	32
4th	1580	8	8

on the computation times for both methods. Figure 8 shows that the cross shape is constrained at the blue nodes and pushed towards the green nodes. Figure 9 shows the final shape of the beam mesh for both the proposed and Pedersen's methods. Tables 5 and 6 show the displacements at force applied nodes (green nodes) using the linear and nonlinear FEMs, respectively.

We conducted fourth experiment with a liver mesh of 465 nodes and 1560 tetrahedral elements. Figure 10 shows that the mesh is constrained at the blue nodes and pulled from the green node (30 N units) in the direction of the arrow. We aimed to observe the similar amount of speedup like previous experiments for a high density mesh. Figure 11 shows the final shape of the beam mesh for both the proposed and Pedersen's methods. Tables 7 and 8 show the displacements at force applied node (node number 271) using the linear and nonlinear FEMs, respectively.

Computation times of the finite element experiments are required to compare how much faster our proposed method is than Pedersen's. When comparing nonlinear FEMs, we calculated the computation times to construct the stiffness matrices as well as the computation times of the nonlinear FEM solutions to determine how different calculations affect them. Table 9 depicts the computation times for the stiffness matrix calculation and Table 10 depicts the computation times for the system solution. Table 11 shows the iteration counts to solve the system using the Newton-Raphson procedure.

The speed-up columns of Tables 9 and 10 depict the speedups of the proposed method compared to Pedersen's method for the stiffness matrix calculation and the system solution using a single thread, respectively. The speedup is calculated as follows:

$$\text{Speedup} = \frac{\text{Runtime (Pedersen's method)}}{\text{Runtime (The proposed method)}}. \tag{40}$$

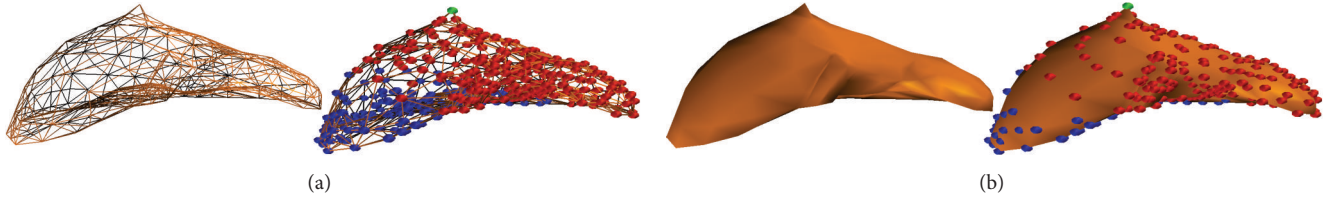(a)   (b)

FIGURE 11: Nonlinear FEM results for both the proposed and Pedersen's methods: (a) left: wireframe surface mesh; right: wireframe surface mesh with nodes; (b) left: shaded mesh; right: shaded mesh with nodes.

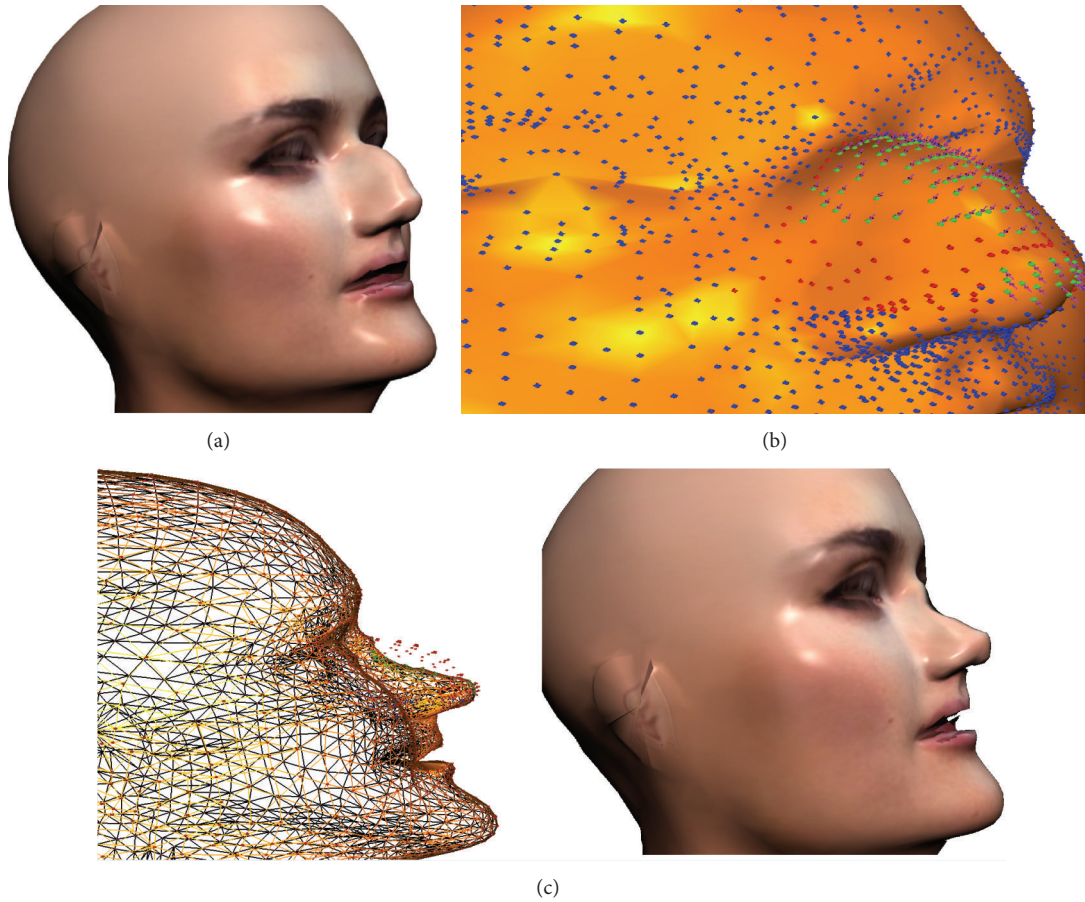(a)   (b)

(c)

FIGURE 12: (a) Initial misshapen nose. (b) The head mesh is constrained at the blue nodes and pushed upwards at the green nodes. (c) The result of the nonlinear FEM: left: wireframe surface mesh with nodes; right: shaded mesh with texture.

Our proposed method outperforms Pedersen's method. On the average, it is 142% faster at computing stiffness matrices because Pedersen's method uses more symbolic terms. However, both methods use Newton-Raphson to solve nonlinear equations, which takes approximately 90% of the computation time. Thus, the overall speedup decreases to 15% on average. In experiment 3, because of more iterations due to root jumps, there was a performance loss against Pedersen's method.

We also applied our method for corrective operation on the misshapen nose of a head mesh. The head mesh is composed of 6709 nodes and 25722 tetrahedral elements (see Figure 12(a)); all the operations were performed in the nose area of only 1458 tetrahedral elements. Figure 12(b) shows that the head mesh is constrained at the blue nodes and pushed upwards at the green nodes and Figure 12(c) shows the result of the nonlinear FEM.

4. Conclusions and Future Work

We propose a new stiffness matrix calculation method for nonlinear FEM that is easier to analyze in terms of constructing elemental stiffness matrices and is faster than Pedersen's method. The proposed method is approximately 2.4 times faster, on average, at computing stiffness matrices and 15% faster at computing the whole system than Pedersen's method.

Although the proposed nonlinear FEM has significant advantages over Pedersen's nonlinear FEM, there is still room for the following development.

(1) Heuristics could be applied to avoid root jumps.

(2) Although we decreased system memory usage by simplifying the solution process for the nonlinear FEM, a significant amount of system memory is still used. The solution process could be further optimized to decrease memory usage.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] K.-J. Bathe, *Finite Element Procedures*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1996.

[2] D. L. Logan, *A First Course in the Finite Element Method*, Cengage Learning, 5th edition, 2012.

[3] P. Pedersen, "Analytical stiffness matrices for tetrahedral elements," *Computer Methods in Applied Mechanics and Engineering*, vol. 196, no. 1-3, pp. 261–278, 2006.

[4] G. R. Joldes, A. Wittek, and K. Miller, "Real-time nonlinear finite element computations on GPU: application to neurosurgical simulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, no. 49–52, pp. 3305–3314, 2010.

[5] Z. A. Taylor, M. Cheng, and S. Ourselin, "High-speed nonlinear finite element analysis for surgical simulation using graphics processing units," *IEEE Transactions on Medical Imaging*, vol. 27, no. 5, pp. 650–663, 2008.

[6] M. Cerrolaza and J. C. Osorio, "Relations among stiffness coefficients of hexahedral 8-noded finite elements: a simple and efficient way to reduce the integration time," *Finite Elements in Analysis and Design*, vol. 55, pp. 1–6, 2012.

[7] J. Bonet and R. D. Wood, *Nonlinear Continu um Mechanics for Finite Element Analysis*, Cambridge University Press, Cambridge, UK, 1st edition, 1997.

[8] C. A. Felippa, "Lecture notes in nonlinear fin ite element methods," Tech. Rep. CUCSSC-96-16, Department of Aerospace Engineering Sciences and Center for Aerospace Structures, University of Colorado, 1996.

[9] P. Pedersen, *The Basic Matrix Approach for Three Simple Finite Elements*, 2008, http://www.topopt.dtu.dk/files/PauliBooks/BasicMatrixApproachInFE.pdf.

[10] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, Frontiers in Applied Mathematics, Society for Industrial Mathematics, 1st edition, 1987.

[11] S. Krenk, *Non-linear Modeling and Analysi s of Solids and Structures*, Cambridge University Press, 1st edition, 2009.

[12] The Mathworks, "Calling MATLAB Engine from C/C++ and Fortran Programs," 2014, http://www.mathworks.com/help/matlab/matlab_external/_f38569.html.

[13] Autodesk, "3ds Max—3D Modeling, Ani mation, and Rendering Software," 2012, http://www.autodesk.com/products/autodesk-3ds-max/.

[14] Maplesoft, "MATLAB Connectivity—Maple Features," 2012, http://www.maplesoft.com/products/maple/features/matlab-connectivity.aspx.

[15] E. Madenci and I. Guven, *The Finite Element Method and Applications in Engineering Using ANSYS*, Springer, Berlin, Germany, 1st edition, 2006.

Submit your manuscripts at
http://www.hindawi.com