An Abaqus user element (UEL) implementation of linear elastostatics

Bibekananda Datta

Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21211, USA

The contents presented in this document are adapted from standard finite element literature (Fish & Belytschko, 2007; Hughes, 1987; Zienkiewicz et al., 2013, 2014). Interested users are suggested to consult textbooks on finite element modeling for solid mechanics, tutorials and textbooks on Fortran programming, and Abaqus documentation (Dassault Systèmes, 2024) for details.

1 Summary of the finite element formulation

The governing partial differential equation for stress equilibrium as well as the boundary conditions in terms of the displacement vector, $\mathbf{g}$, on Γ_g and the Cauchy traction, $\mathbf{t}$, on Γ_t are given by,

$$\begin{aligned} \operatorname{div}(\boldsymbol{\sigma}) + \rho\mathbf{b} &= 0 \qquad && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g} && \text{on } \Gamma_g, \\ \boldsymbol{\sigma}\cdot\mathbf{n} &= \mathbf{t} && \text{on } \Gamma_t, \end{aligned} \tag{1.1}$$

where, $\mathbf{b}$ is the body force per mass.

The global weak form, $\mathcal{W}_{\mathbf{u}}(\mathbf{u})$, can be written as,

$$\mathcal{W}_{\mathbf{u}}(\mathbf{u}) = -\int_{\Omega} \boldsymbol{\sigma}\left(\boldsymbol{\varepsilon}(\mathbf{u})\right) : \operatorname{grad}(\mathbf{w})\, dv + \int_{\Omega} \rho\mathbf{b}\cdot\mathbf{w}\, dv + \int_{\Gamma_t} \mathbf{t}\cdot\mathbf{w}\, ds = 0, \tag{1.2}$$

where, $\mathbf{u} \in \mathscr{U}$ be the trial solution which satisfies $u_i = g_i$, and $\mathbf{w} \in \mathscr{W}$ be a vector test (or weight) function which satisfies $w_i = 0$ on Γ_g, and for small deformation, the infinitesimal strain tensor can be defined as,

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left[\operatorname{grad}(\mathbf{u}) + (\operatorname{grad}(\mathbf{u}))^{\top}\right] = \operatorname{sym}[\operatorname{grad}(\mathbf{u})], \tag{1.3}$$

and the constitutive relation for isotropic linear elastic material is given by,

$$\boldsymbol{\sigma} = \mathbb{c} : \boldsymbol{\varepsilon}, \qquad \Rightarrow \boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\varepsilon})\mathbb{1} + 2\mu\boldsymbol{\varepsilon}, \qquad \Rightarrow \sigma_{ij} = \mathbb{c}_{ijkl}\varepsilon_{kl} = \lambda\varepsilon_{kk}\delta_{ij} + 2\mu\varepsilon_{ij}, \tag{1.4}$$

where, $\boldsymbol{\sigma}$ is known as the Cauchy stress. Here, $\mathbb{c}$ is known as the fourth-order elasticity or modulus tensor, and μ and λ are known as Lamé constants. In finite element implementation, the constitutive relations are reduced to a matrix-vector form using the Voigt notation convention.

Using the standard Galerkin discretization, the discretized residual for each element, $\mathbf{R}^e_{\mathbf{u}}(\mathbf{u}_e)$, can be written as,

$$\mathbf{R}^e_{\mathbf{u}}(\mathbf{u}_e) = -\int_{\Omega^e} \mathbf{B}^\top_{\mathbf{u}} \boldsymbol{\sigma}\left(\boldsymbol{\varepsilon}(\mathbf{u}_e)\right)\, dv + \int_{\Omega^e} \rho \mathbf{N}^\top_{\mathbf{u}} \mathbf{b}\, dv + \int_{\Gamma^e_t} \mathbf{N}^\top_{\mathbf{u}} \mathbf{t}^e\, ds = 0, \tag{1.5}$$

where, $[\mathbf{N}^a_{\mathbf{u}}]$ is the nodal matrix and $[\mathbf{N}_{\mathbf{u}}]$ is the element matrix of the interpolation functions, and for a three-dimensional case, they are given by,

$$\mathbf{N}^a_{\mathbf{u}} = \begin{bmatrix} N_i & 0 & 0 \\ 0 & N_i & 0 \\ 0 & 0 & N_i \end{bmatrix}, \quad \mathbf{N}_{\mathbf{u}} = \begin{bmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \cdots & N_{\text{nen}} & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \cdots & 0 & N_{\text{nen}} & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \cdots & 0 & 0 & N_{\text{nen}} \end{bmatrix}, \tag{1.6}$$

and the strain-displacement matrix, *i.e.*, the symmetric gradient matrix at the element level, $[\mathbf{B}_{\mathbf{u}}]$, and at the node level, $[\mathbf{B}^a_{\mathbf{u}}]$, are given by,

$$\mathbf{B}_{\mathbf{u}} = \begin{bmatrix} \mathbf{B}^1_{\mathbf{u}} & \mathbf{B}^2_{\mathbf{u}} & \mathbf{B}^3_{\mathbf{u}} & \cdots & \cdots & \mathbf{B}^{\text{nen}}_{\mathbf{u}} \end{bmatrix}, \quad \text{and} \quad \mathbf{B}^a_{\mathbf{u}} = \begin{bmatrix} N^a_{,1} & 0 & 0 \\ 0 & N^a_{,2} & 0 \\ 0 & 0 & N^a_{,3} \\ 0 & N^a_{,3} & N^a_{,2} \\ N^a_{,3} & 0 & N^a_{,1} \\ N^a_{,2} & N^a_{,1} & 0 \end{bmatrix}. \tag{1.7}$$

Now, the element stiffness matrix, $\mathbf{k}^e_{\mathbf{uu}}$, is given by,

$$\mathbf{k}^e_{\mathbf{uu}} = -\frac{\partial \mathbf{R}^e_{\mathbf{u}}}{\partial \mathbf{u}_e} = \int_{\Omega^e} \mathbf{B}^\top_{\mathbf{u}} \mathbf{D} \mathbf{B}_{\mathbf{u}}\, dv \tag{1.8}$$

where $\mathbf{D}$ represents the Voigt matrix form of the elasticity tensor which is also referred to as the stiffness matrix of the material in the context of the linear elasticity.

Corresponding to the interpolation function matrix, $[\mathbf{N}_{\mathbf{u}}]$, and strain-displacement matrix, $[\mathbf{B}_{\mathbf{u}}]$, the degrees of freedom of an element, $\mathbf{u}_e$, and stress components in Voigt vector form, $\boldsymbol{\sigma}$, are represented in the following manner.

$$\begin{gathered} \mathbf{u}_e = \begin{bmatrix} u^1_1 & u^1_2 & u^1_3 & u^2_1 & u^2_2 & u^2_3 & \cdots & \cdots & \cdots & u^{\text{nen}}_1 & u^{\text{nen}}_2 & u^{\text{nen}}_3 \end{bmatrix}^\top_{\text{nsd}*\text{nen}\times 1}, \\ \text{and } \boldsymbol{\sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{23} & \sigma_{13} & \sigma_{12} \end{bmatrix}^\top \end{gathered} \tag{1.9}$$

For two-dimensional cases, the dimensions of the interpolation function matrix, $[\mathbf{N}_{\mathbf{u}}]$, strain-

displacement matrix, $[\mathbf{B_u}]$, nodal degrees of freedom, $\mathbf{u}_e$, Voigt stress vector, $\boldsymbol{\sigma}$, material stiffness matrix $[\mathbf{D}]$ are needed to be reduced accordingly.

A simplified pseudo-code or procedure for implementing the finite element model through the Abaqus `UEL(...)` subroutine is given below.

Procedure 1: Small strain UEL subroutine implementation in Abaqus/Standard

---

Input : `PROPS, COORDS, JELEM, JTYPE, NNODE, NDOFEL, TIME, DTIME, U, DU, V, A, PREDEF, JDLTYP, NDLOAD, MDLOAD, DDLMAG, ALMAG`

Output : `AMATRX, RHS, PNEWDT, ENERGY, SVARS`

Get `nInt` $\leftarrow$ `PROPS` and `nDim`, `nTens` $\leftarrow$ `JTYPE`
Initialize $\{\mathbf{N}^a_{\mathbf{u}}, \mathbf{B}^a_{\mathbf{u}}, \mathbf{N}_{\mathbf{u}}, \mathbf{B}_{\mathbf{u}}, \mathbf{k}^e_{\mathbf{uu}}\} = 0$
Get nodal displacement vector of the element, $\mathbf{u}_e, \Delta\mathbf{u}_e$
Get $w_{\text{int}}, \boldsymbol{\xi}_{\text{int}} \leftarrow$ SUBROUTINE `gaussQuadrtr`(`nDim`, `nNode`)
for $k = 1$ **to** `nInt` **do**
    Get $[\mathbf{N_u}], \left[\frac{\partial \mathbf{N_u}}{\partial \boldsymbol{\xi}}\right] \leftarrow$ SUBROUTINE `interpFunc`(`nDim`, `nNode`)
    Calculate $\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}} = [\overline{\mathbf{x}}_e]\left[\frac{\partial \mathbf{N_u}}{\partial \boldsymbol{\xi}}\right]$
    Calculate $\frac{\partial \mathbf{N_u}}{\partial \mathbf{x}} = \frac{\partial \mathbf{N_u}}{\partial \boldsymbol{\xi}}\left(\frac{\partial \mathbf{x}}{\partial \boldsymbol{\xi}}\right)^{-1}$ and J_ξ
    for $i = 1$ **to** `nNode` **do**
        Form $\mathbf{N}^a_{\mathbf{u}}$(`nDim`, `NDOFEL`) and $\mathbf{B}^a_{\mathbf{u}}$(`nTens`, `nDim`) matrix
        $\mathbf{N_u}(1:\texttt{nDim}, \texttt{nDim}*(\texttt{i}-1)+1:\texttt{nDim}*\texttt{i}) = \mathbf{N}^a_{\mathbf{u}}(1:\texttt{nDim}, 1:\texttt{nDim})$
        $\mathbf{B_u}(1:\texttt{nTens}, \texttt{nDim}*(\texttt{i}-1)+1:\texttt{nDim}*\texttt{i}) = \mathbf{B}^a_{\mathbf{u}}(1:\texttt{nTens}, 1:\texttt{nDim})$
    end
                                                    // end of nodal loop
    Calculate $\boldsymbol{\varepsilon} = \mathbf{B_u}\mathbf{u}_e$
    Get $\boldsymbol{\sigma}, \mathbf{D} \leftarrow$ SUBROUTINE `umatElastic`(`PROPS`, $\boldsymbol{\varepsilon}$) // UMAT returns Cauchy stress and material tangent in Voigt form
    $\mathbf{k}^e_{\mathbf{uu}} = \mathbf{k}^e_{\mathbf{uu}} + w_{\text{int}}(k)\det(J_\xi)\left(\mathbf{B}^\top_{\mathbf{u}}\mathbf{D}\mathbf{B_u}\right)$
    $\mathbf{R}^e_{\mathbf{u}} = \mathbf{R}^e_{\mathbf{u}} - w_{\text{int}}(k)\det(J_\xi)\left(\mathbf{B}^\top_{\mathbf{u}}\boldsymbol{\sigma} - \rho\mathbf{N}^\top_{\mathbf{u}}\mathbf{b}\right)$
end
                                                    // end of integration point loop
Assign `AMATRX` $= \mathbf{k}^e_{\mathbf{uu}}$ and `RHS` $= \mathbf{R}^e_{\mathbf{u}}$

The current implementation includes 3D continuum solid elements and plane-strain elements, but body force and traction loading were not included. Since Abaqus does not natively support viewing element output from the user element, a layer of additional dummy elements with `UVARM(...)` subroutine from Abaqus was used to view the results.

2 Modeling in Abaqus

If and only if the user element has the same topology as any standard built-in element available in Abaqus, we can build the primary model in Abaqus/CAE, export the input file (`.inp`), and modify it to execute with Abaqus/Standard solver. Building the primary model is a standard exercise, and it is strongly recommended to check the option **Do not use parts and assemblies in input files** from the **model attributes** drop-down menu before exporting the input file of a model. Once the input is available, it must be modified following the instructions in the Abaqus user manual (Dassault Systèmes, 2024).

The first step is to include the element definition for the user elements in the input file by removing the standard element defined with `*Element` keyword in generating the model.

```
*User Element, Type=< >, Nodes=< >, Coordinates=< >, Properties=<
   >, Iproperties=< >
<list of degrees of freedom>
*Element, type=< >
<list of element nodal connectivity>
```

The second step is to define the properties of the user elements by element sets by removing standard keywords `*Section` and `*Material`.

```
*Uel property, Elset= < >
<list of properties>
```

The third step is optional, but if the user wants to visualize the results in Abaqus/Viewer, a set of overlaying dummy elements needs to be added to the input file.

```
*Element, Type=< >
<list of element nodal connectivity>
*Elset, Elset=elDummy
< >
*Solid section, Elset=elDummy, Material=Dummy
*Material, Name=Dummy
*Elastic
1.e-20
*User output variables
< >
```

The fourth step is to request element-level output in load steps.

```
*Element output, Elset=< >
uvarm
```

3 Source code availability

GitHub repository: https://github.com/bibekanandadatta/Abaqus-UEL-Elasticity.

4 Citation

Datta, B. (2024). *An Abaqus user element (UEL) implementation of linear elastostatics*. Zenodo. https://doi.org/10.5281/zenodo.11075088 (APA format)

Bibliographic references

Dassault Systèmes. (2024). *SIMULIA User Assistance 2024: Abaqus*. https://help.3ds.com

Fish, J., & Belytschko, T. (2007). *A First Course in Finite Elements*. John Wiley & Sons, Inc.

Hughes, T. J. R. (1987). *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis* (1st ed.). Prentice-Hall.

Zienkiewicz, O. C., Taylor, R. L., & Fox, D. (2014). *The Finite Element Method for Solid and Structural Mechanics* (7th ed.). Butterworth-Heinemann.

Zienkiewicz, O. C., Taylor, R. L., & Zhu, J. Z. (2013). *The Finite Element Method: Its Basis and Fundamentals* (7th ed.). Butterworth-Heinemann.