Lehrstuhl für Angewandte Mechanik
Technische Universität München

Yifei Wang

Implementation and Investigation of Stress-Recovery-Algorithms in the Finite Element Toolbox AMfe

Semesterarbeit

30. 05. 2016

Supervisor:
Johannes Rutzmoser

Abstract

The motivation of this paper is to implement Python code in Finite Element Toolbox AMfe for the purpose of achieving an relatively exact approximation of strain and stress. The assignment we are trying to complete in this paper is the expansion of Finite Element Toolbox AMfe, which will make it possible to solve strain and stress with 2D element types, namely Tri3, Tri6, Quad4, Quad8 and 3D element types, namely Tet4 and Tet10. The approach we adopted to solve the problem is Stress-Recovery-Algorithms. This is a method based on extrapolation from Gauss points in contrast to the Direct-Evaluation-Algorithms. The objects of my analysis for stress recovery in this research are six element types: quadrilateral element with four nodes (Quad4), quadrilateral element with eight nodes (Quad8), triangular element with three nodes (Tri3), triangular element with six nodes (Tri6), tetrahedral element with four nodes (Tet4), and tetrahedral element with four node (Tet10). Patch test for both algorithms and a convergence analysis for all six element types are also presented in this paper. Towards the end of this paper, the strain and stress contour plot by both algorithms(Stress-Recovery-Algorithm and Direct-Evaluation-Algorithms) will be compared to the solution calculated by the commercial Finite Element software ANSYS. The significance of our obtained results is that the solution of Stress-Recovery-Algorithms is closer to the analytical solution than the solution of Direct-Evaluation-Algorithms for Tri6, Quad4, Quad8, and Tet10. For both element types Tri3 and Quad4, the respective results for nodal stress from the Stress-Recovery-Algorithms and the Direct-Evaluation-Algorithms are exactly the same.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Johannes Rutzmoser for the continuous support of my study and related research, for his patience, motivation, and immense knowledge I am deeply grateful of his help in the completion of this thesis. His guidance helped me in all the time of finite element research and get in with Finite- Element- Research-code. I could not have imagined having a better advisor and mentor for my study in Finite-Element- Method.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The Finite Element Method (FEM) is a numerical technique for finding approximate solutions to boundary value problems for partial differential equations. FEM contains linear FEM and non-linear FEM. Linear analysis follows the equation $K \cdot D = F$, in which K stands for stiffness matrix, D stands for displacement, and F stands for force. It means that the correlation of force and displacement is linear. This relation between force and displacement is only valid for materials that have elastic linear property. But in real situation, elastic materials such as steel has also non- linear region. (see the Figure 1.1). The property of steel is elastic linear up until yield point. Then the steel is yielding and becomes non-linear. Non-linear states include geometry non-linear and material non-linear. The example in Figure 1.1 describes a non-linear case and it also the main topic of this thesis. In the fields of engineering and science, FEM is a powerful tool in producing strength visualization and can help engineers to minimize weight, materials and costs. The accuracy of solution are only limited by the quality of model and by the available computational power. Ever since the computational power has been improved enormously, the FEM software has offered a wide range of simulations of complex model designs and system analyses.
AMfe is a nonlinear finite element code for structural application at the chair of applied mechanics in Technical University of Munich. AMfe Toolbox is developed in Python and Fortran. Python is a high-level, interpreted, and dynamically-typed programming language. There are many numerical packages built on Python, such as Numpy, Scipy, Pandas. These packages provide high-performance and easy to use data structures, both of which match the FEM developing work. The fact that Python is a high-level programming language makes it less time-consuming to develop the code. However, Python is slow for repeated execution of low-level task. Each Python operation comes with a small type-checking overhead, and when many repeated small operations add up, the overhead becomes significant. For this reason, A part of the AMfe code is rewritten in Fortran. Fortran is a general-purpose and imperative programming language that is specially tailored for numeric computation and scientific computing. Therefore, the combination of Python and Fortran retains the advantages of both - the easy-to-develop nature of Python and the fast numeric computation in Fortran. The aim of AMfe Toolbox is to solve and anal-

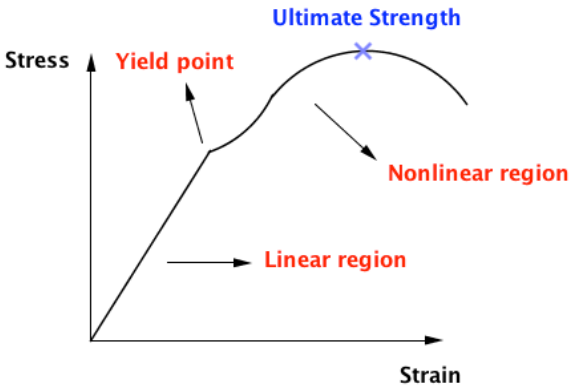

Figure 1.1: Stress-Strain curve for steel

yse FEM problems, especially structural mechanical problems. The toolbox contains several modules with different functions to solve problems step by step. A simple structure of AMfe is depicted in Figure 1.3. Displacement, stress, and strain are three important factors in structural mechanical engineering. The function for calculating nodal displacements is completed in AMfe Toolbox, but functions for calculating nodal stress and strain remain blank. Strain and stress calculations are important in structural mechanical analysis and geometric design. For this reason, functions for solving of stress and strain will be added to AMfe Toolbox. When strain and stress calculations are completed, the aim of research is then shifted to improving the accuracy of these results. In order to do so, stress recovery is an approach to extrapolate the element solution to nodal solution. The goal is to be as accurate as possible in the computed displacements while keeping the computational effort reasonable. When the most important functions are completed and running well, it is time to check if AMfe Toolbox exports the reasonable results. The results of stress and strain from AMfe Toolbox will be compared with a commercial FEM software - ANSYS. The comparison of each element type will be recorded and discussed.

1.2 Structure of Study

This study can be divided into three parts. Figure 1.2 shows the structural flow chart in this study. The first part is marked with a green box. The main function for this part is to export the data of a meshed geometry using ANSYS Parametric Design Language(APDL) code and import the data into the AMfe Toolbox. This procedure, which is called Pre-Processing, is the first step in solving a problem in Finite Element Analysis. This step also ensures the analyses from both ANSYS and AMfe Toolbox have the same set of elements and nodes. The processing step comes next, which is marked with a yellow box. AMfe Toolbox has several modules with different functions. These modules combined provides all calculation functions needed for processing. This study is focusing on calculating stress and strain. The last part marked by blue is to check the accuracy of strain and stress using different types of elements.

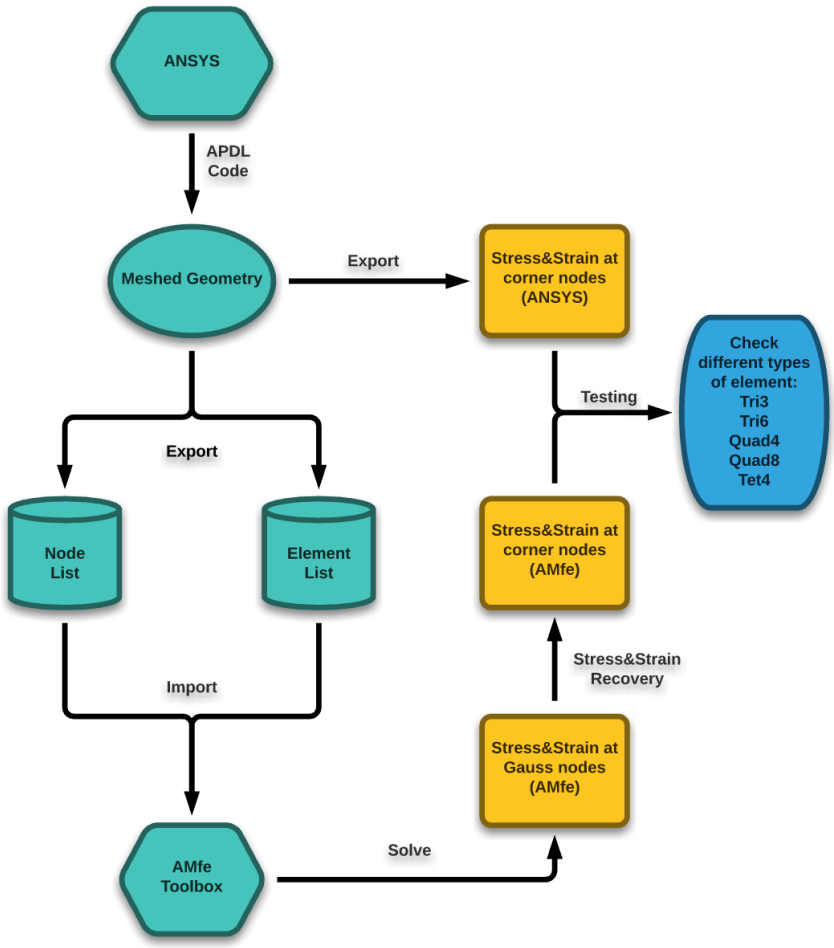

Figure 1.2: Flow chart of structure

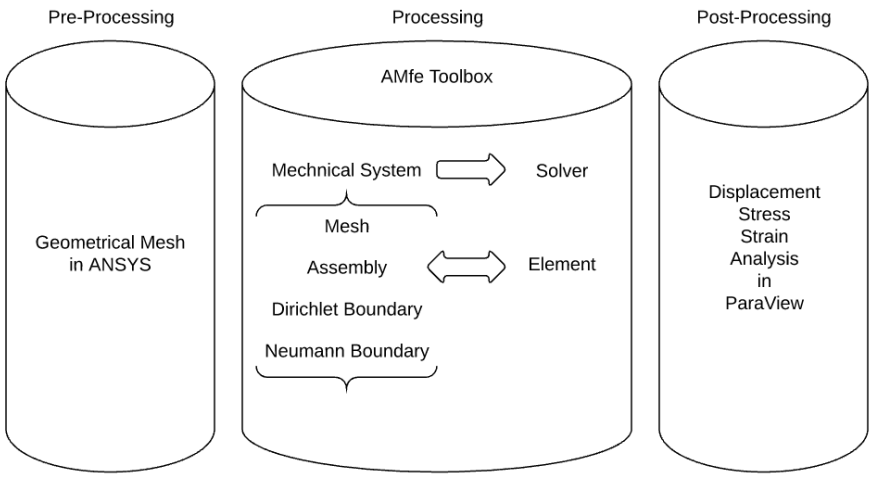

Figure 1.3: Overview of this essay structure

Chapter 2

Numerical Aspect for Finite Element Formulation

2.1 Shape Function

2.1.1 Quadrilateral Elements

The goal of the AMfe Code in the structural mechanical field is to get the approximate solutions to position, displacement, stress, and strain. According to the Finite Element theory of W. A. Wall [Wall 2014 p.75], the approximation is dependent on the shape function N. The standard approach to define the shape functions is choosing them as simple polynomials that are associated to nodes. There are n nodes and n shape functions in one single element. The i-th shape function takes the value of 1 at the i-th node in reference coordinates, then the j-th shape function can be written like this:

$$\begin{aligned} &\text{if } j = i \text{ then } N_j\left(\hat{\xi}_i\right) = 1 \\ &\text{if } j \neq i \text{ then } N_j\left(\hat{\xi}_i\right) = 0 \end{aligned}$$

for $i, j = 1,...,n$.

In Figure 2.1, linear and quadratic shape functions are expressed together with corresponding node positions. When referring to an element with three nodes, as shown in the right of Figure 2.1, there are three restrictions for each shape functions:

$$\begin{aligned} &N_1\left(\hat{\xi}_1 = -1\right) = 1,\ N_1\left(\hat{\xi}_2 = 0\right) = 0,\ N_1\left(\hat{\xi}_3 = 1\right) = 0; \\ &N_2\left(\hat{\xi}_1 = -1\right) = 0,\ N_2\left(\hat{\xi}_2 = 0\right) = 1,\ N_2\left(\hat{\xi}_3 = 1\right) = 0; \\ &N_3\left(\hat{\xi}_1 = -1\right) = 0,\ N_3\left(\hat{\xi}_2 = 0\right) = 0,\ N_3\left(\hat{\xi}_3 = 1\right) = 1 \end{aligned}$$

For quadratic polynomial, the shape function has three coefficients, and it can be constructed as follows:

$$p_{quad}\left(\xi\right) = c_0 + c_1\xi + c_2\xi^2$$

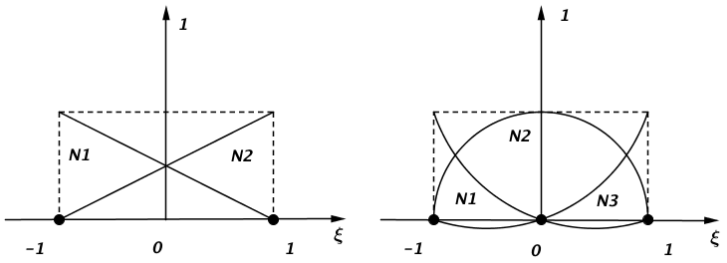

Figure 2.1: Linear shape function and quadratic shape function.

According to the work of W. A. Wall [Wall 2014 p. 75], Lagrange polynomials are a general group of polynomials with the node undermentioned association property: a Lagrange polynomial $l_k^n - 1$ of order $n-1$ transits n nodes with coordinates $\bar{\xi}^j$ $(j = 1, ..., n)$ of which the single node k equals one $\left(l_k^{n-1}\left(\bar{\xi}^k\right) = 1\right)$ and every other node equals zero $\left(l_k^{n-1}\left(\bar{\xi}^k\right) = 0 \quad \text{for all} \quad j \neq k\right)$.

$$l_k^{n-1}(\xi) = \prod_{j=1, j\neq k}^{n} \frac{\xi - \bar{\xi}^1}{\bar{\xi}^k - \bar{\xi}^j} = \frac{(\xi-\bar{\xi}^1)\cdots(\xi-\bar{\xi}^{k-1})(\xi-\bar{\xi}^{k+1})\cdots(\xi-\bar{\xi}^n)}{(\bar{\xi}^k-\bar{\xi}^1)\cdots(\bar{\xi}^k-\bar{\xi}^{k-1})(\bar{\xi}^k-\bar{\xi}^{k+1})\cdots(\bar{\xi}^k-\bar{\xi}^n)}$$

$$k = 1, 2...., n$$

For instance, to calculate linear, i.e. order 1, Lagrange polynomials can be calculated by $n-1 = 1 \Rightarrow n = 2$. Using $\bar{\xi}^1 = -1$ and $\bar{\xi}^2 = +1$ we can get the shape functions as: $l_1^1(\xi) = -1/2(\xi - 1)$ and $l_2^1(\xi) = 1/2(\xi + 1)$. Then the linear shape functions can be simply constructed as $N_1 = l_1^1 = 1/2(1 - \xi)$ and $N_2 = l_2^1 = 1/2(1 + \xi)$.

2.1.2 Triangular and Tetrahedral Elements

According to the work of Carlos A. Felippa [Felippa 2004 p.79], triangular and tetrahedral elements have higher flexibility than that of quadrilateral and hexahedral elements in meshing complex geometries, thus the former is often preferred to the latter. The shape functions can be solved by an analogous method, but triangular and tetrahedral elements are expressed in area and volume coordinates, respectively. Figure 2.2 shows the geometries and shape functions. A visualization of the coordinates representing area and volume fractions is expressed below:

$$L_1 = \frac{\text{area}P23}{\text{area}123}, \qquad V_1 = \frac{\text{area}P234}{\text{area}1234}$$

$$\sum L_i = 1 \qquad \sum V_i = 1$$

2.2 Non-linear Element Formulation

This section provides an introduction to the most common model and simulation technique for both 2D and 3D solid bodies in Non-Finite-Element-Method. The following section is based on

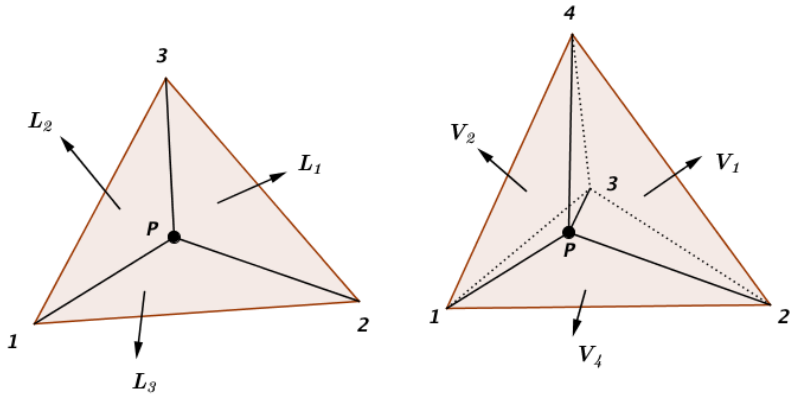

Figure 2.2: triangular and tetradral elemnets.

the work of Johaness Rutzmoser [Rutzmoser 2016 p.1-5] in non-linear element formulation. In the first part of this introduction, we will pick a fictional 3D element with five nodes as our object of study. The coordinate system is based on three coordinate, namely ξ_1, ξ_2 ,and ξ_3. The shape function of this 3D element can be ordered in Voigt notation as follows:

$$\boldsymbol{N}(\xi) = \begin{pmatrix} N_1(\xi) \\ N_2(\xi) \\ N_3(\xi) \\ N_4(\xi) \\ N_5(\xi) \end{pmatrix} \tag{2.1}$$

We can denote the coordinate of this element as vector $\boldsymbol{X}^e$. The columns of $\boldsymbol{X}^e$ stands for its axis direction and the rows of $\boldsymbol{X}^e$ stands for the index of nodes. The matrix of $\boldsymbol{X}^e$ is shown as:

$$\boldsymbol{X}^e = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \\ X_4 & Y_4 & Z_5 \\ X_5 & Y_5 & Z_6 \end{pmatrix} \tag{2.2}$$

The coordinates $\boldsymbol{X}$ of this element in the initial configuration relies on the local coordinate of the element ξ and the shape function $\boldsymbol{N}$. It can be described as:

$$\boldsymbol{X}(\xi) = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \left(\boldsymbol{X}^e\right)^T N = \begin{pmatrix} X_1 & X_2 & X_3 & X_4 & X_5 \\ Y_1 & Y_2 & Y_3 & Y_4 & Y_5 \\ Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{pmatrix} \tag{2.3}$$

The displacements $\boldsymbol{u}$ at any points of the element are interpolated from the nodal coordinate, in the same manner that was completed for the coordinate $\boldsymbol{X}$ above. Then the displacement field

can be shown as:

$$\boldsymbol{u}(\xi) = \begin{pmatrix} u_x(\xi) \\ u_y(\xi) \\ u_z(\xi) \end{pmatrix} = (\boldsymbol{u}^e)^T N = \begin{pmatrix} u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} \\ u_{z1} & u_{z2} & u_{z3} & u_{z4} & u_{z5} \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{pmatrix} \tag{2.4}$$

This approach is called the isoparametric concept, which makes it possible to present the parameters at any given point within an element. The magnitude of parameters at a one of such points can be calculated through both the known values of parameters and shape functions at each node. An example of displacements can be formed like this:

$$\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = (\boldsymbol{u}^e)^T \frac{\partial \boldsymbol{N}}{\partial \boldsymbol{X}} = (\boldsymbol{u}^e)^T \tilde{\boldsymbol{B}}_0 \tag{2.5}$$

The deviation of a parameter can be passed to the shape function. Through this approach, we can define a new term $\tilde{\boldsymbol{B}}_0 = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}}$. The expansion of $\tilde{\boldsymbol{B}}_0$ can be derived as:

$$\frac{\partial \boldsymbol{N}}{\partial \boldsymbol{X}} = \tilde{\boldsymbol{B}}_0 = \begin{pmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial Z} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial Z} \\ \frac{\partial N_3}{\partial X} & \frac{\partial N_3}{\partial Y} & \frac{\partial N_3}{\partial Z} \\ \frac{\partial N_4}{\partial X} & \frac{\partial N_4}{\partial Y} & \frac{\partial N_4}{\partial Z} \\ \frac{\partial N_5}{\partial X} & \frac{\partial N_5}{\partial Y} & \frac{\partial N_5}{\partial Z} \end{pmatrix} = \frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\xi}} \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}} \tag{2.6}$$

The first term $\frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\xi}}$ in Equation 2.6 can be directly expressed with shape function relating coordinate ξ. It is shown as:

$$\frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\xi}} = \begin{pmatrix} \frac{\partial N_1}{\partial \xi_1} & \frac{\partial N_1}{\partial \xi_2} & \frac{\partial N_1}{\partial \xi_3} \\ \frac{\partial N_2}{\partial \xi_1} & \frac{\partial N_2}{\partial \xi_2} & \frac{\partial N_2}{\partial \xi_3} \\ \frac{\partial N_3}{\partial \xi_1} & \frac{\partial N_3}{\partial \xi_2} & \frac{\partial N_3}{\partial \xi_3} \\ \frac{\partial N_4}{\partial \xi_1} & \frac{\partial N_4}{\partial \xi_2} & \frac{\partial N_4}{\partial \xi_3} \\ \frac{\partial N_5}{\partial \xi_1} & \frac{\partial N_5}{\partial \xi_2} & \frac{\partial N_5}{\partial \xi_3} \end{pmatrix} \tag{2.7}$$

The second term $\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}}$ is not straightforward to obtain. However, it is much easier to get the matrix $\frac{\partial \boldsymbol{X}}{\partial \boldsymbol{\xi}}$. This matrix is called Jacobian matrix $\boldsymbol{J}$, which is a matrix of $\boldsymbol{\xi}$ with respect to x. The inverse Jacobian matrix is denoted as $\boldsymbol{J}^{-1}$, which is the second term in the Equation 2.6.
The deformation gradient $\boldsymbol{F}$ describes the mapping of an infinitesimal fiber from its initial state to its new position in the current configuration. $\boldsymbol{F} = \boldsymbol{H} + \boldsymbol{I}$ can be derived through the transient matrix $\boldsymbol{H} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = (\boldsymbol{u}^e)^T \tilde{\boldsymbol{B}}_0$. The Green-Lagrange strain tensor can also be written as:

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{H} + \boldsymbol{H}^T + \boldsymbol{H}^T\boldsymbol{H}\right) \tag{2.8}$$

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I}\right) \tag{2.9}$$

The transient matrix $\boldsymbol{H}$ can be expressed in more details as:

$$\boldsymbol{H} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{X}} = (\boldsymbol{u}^e)^T \frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\xi}}\left[\left(\boldsymbol{X}^e\right)^T \frac{\partial \boldsymbol{N}}{\partial \boldsymbol{\xi}}\right]^{-1} \tag{2.10}$$

When we have result for strain $\boldsymbol{E}$, It is possible to calculate stress $\boldsymbol{S}$. One simply constitutive equation between stress $\boldsymbol{S}$ (2. Piola-Kirchhoff-Stress tensor) and $\boldsymbol{E}$ (Green-Lagrange-Strain tensor) can be formulated as:

$$\boldsymbol{S} = \boldsymbol{C} : \boldsymbol{E} \tag{2.11}$$

The value of stress is subsequently transferred into the degrees of freedom(DOFs) at each nodes. The principle for stress calculation can be formulated as follows:
In the Total Lagrange approach, the principle of virtual work represents that the internal strain

energy $\delta \boldsymbol{W}_{int} = \int \boldsymbol{\sigma} : \delta\boldsymbol{\epsilon} \mathrm{d}\Omega_0 = \int \boldsymbol{S} : \delta\boldsymbol{E} \mathrm{d}\Omega_0$ equals the external energy $\delta \boldsymbol{W}_{ext} = (\delta \boldsymbol{u}^{e,\upsilon})^T \boldsymbol{f}^{\upsilon}_{nl}$, which is caused by external nodal force. The complete equation can be expressed as:

$$\delta \boldsymbol{W} = \delta \boldsymbol{W}_{ext} = (\delta \boldsymbol{u}^{e,\upsilon})^T \boldsymbol{f}^{\upsilon}_{int} = \int \boldsymbol{S} : \delta\boldsymbol{E} \mathrm{d}\Omega_0 = \int \left(\delta \boldsymbol{E}^{\upsilon}\right)^T \boldsymbol{S}^{\upsilon} \Omega_0 \tag{2.12}$$

The internal deformation energy can be computed by matrix-vector-product in voigt-notation or direct product of two matrix with notation (:). Now, we evaluate the variation of Green-Lagrange strain tensor $\delta\boldsymbol{E}$. From equation 2.9, it is obvious that the variation of tensor $\delta\boldsymbol{E}$ is determined by the variation of deformation gradient. According to above equation 2.10, the variation of deformation gradient can be transformed into:

$$\delta \boldsymbol{F} = \delta \boldsymbol{H} = (\delta \boldsymbol{u}^{e})^T \frac{\delta \boldsymbol{N}}{\delta \boldsymbol{X}} = (\delta \boldsymbol{u}^{e})^T \tilde{\boldsymbol{B}_0} \tag{2.13}$$

Consequently, the variation of tensor $\delta\boldsymbol{E}$ is:

$$\delta \boldsymbol{E} = \frac{1}{2} \left(\delta \boldsymbol{F}^T \boldsymbol{F} + \boldsymbol{F}^T \delta F\right) \tag{2.14}$$

Green-Lagrange-Strain $\delta\boldsymbol{E}$ is often represented in a complex matrix form, thus it is often expressed in Voigt-Notation in computer programming. Then, $\delta\boldsymbol{E}^{\upsilon}$ can be coupled with $\boldsymbol{B}_0$-matrix as:

$$\delta \boldsymbol{E}^{\upsilon} = \boldsymbol{B}_0 \delta \boldsymbol{u}^{e,\upsilon} \tag{2.15}$$

The entry of $\boldsymbol{B}_0$ will be packed into a black box and directly implemented in the FEM simulation. By solving simultaneous equations of Equation 2.12 and Equation 2.15, the non-linear force $\boldsymbol{f}^{\upsilon}_{int}$ results in:

$$\delta \boldsymbol{W} = (\delta \boldsymbol{u}^{e,\upsilon})^T \boldsymbol{f}^{\upsilon}_{int} = \int \left(\delta \boldsymbol{E}^{\upsilon}\right)^T \boldsymbol{S}^{\upsilon} \mathrm{d}\Omega_0 = (\delta \boldsymbol{u}^{e,\upsilon})^T \int \boldsymbol{B}_0^T \boldsymbol{S}^{\upsilon} \mathrm{d}\Omega_0 \tag{2.16}$$

$$\boldsymbol{f}^{\upsilon}_{int} = \int \boldsymbol{B}_0^T \boldsymbol{S}^{\upsilon} \mathrm{d}\Omega_0 \tag{2.17}$$

The non-linear internal force of element concerning the coordinate of node can be integrated by $\boldsymbol{B}_0$-martrix with second Piola-Kirchhoff-Stress tensor in Voigt-Notation. To obtain the tangential stiffness matrix, it is necessary to calculate the partial derivative of internal force $\boldsymbol{f}^{\upsilon}_{int}$ in respect to the nodal degrees of freedom(DOFs) $\boldsymbol{u}^{e,\upsilon}$:

$$\frac{\partial \boldsymbol{f}^{\upsilon}_{int}}{\partial \boldsymbol{u}^{e,\upsilon}} = \boldsymbol{K} = \int \frac{\partial \boldsymbol{B}_0^T}{\partial \boldsymbol{u}^{e,\upsilon}} \boldsymbol{S}^{\upsilon} \mathrm{d}\Omega_0 + \int \boldsymbol{B}_0^T \frac{\partial \boldsymbol{S}^{\upsilon}}{\partial \boldsymbol{u}^{e,\upsilon}} \mathrm{d}\Omega_0 = \boldsymbol{K}_{geo} + \boldsymbol{K}_{mat} \tag{2.18}$$

Stiffness matrix can be divided into two terms, one of the terms is the material stiffness matrix $\boldsymbol{K}_{amt}$, which is shown as:

$$\boldsymbol{K}_{mat} = \int \boldsymbol{B}_0^T \frac{\partial \boldsymbol{S}^{\upsilon}}{\partial \boldsymbol{u}^{e,\upsilon}} \mathrm{d}\Omega_0 = \int \boldsymbol{B}_0^T \frac{\partial \boldsymbol{S}^{\upsilon}}{\partial \boldsymbol{E}^{\upsilon}} \frac{\partial \boldsymbol{E}^{\upsilon}}{\partial \boldsymbol{u}^{e,\upsilon}} \mathrm{d}\Omega_0 = \int \boldsymbol{B}_0^T \boldsymbol{C}^{SE} \boldsymbol{B}_0 \mathrm{d}\Omega_0 \tag{2.19}$$

The derivative of stress in respect to strain is considered as Tangent-Modulus $\boldsymbol{C}^{SE} = \frac{\partial \boldsymbol{S}^v}{\partial \boldsymbol{E}^v}$ by constitutive equation. The derivative of strain is $\boldsymbol{B}_0$ as determined in equation 2.9.
The other term, the geometrical stiffness matrix, is much more complex to derive. In short, based on the relation between deformation gradient and continuum mechanics $\boldsymbol{P} = \boldsymbol{S}\boldsymbol{F}^T$, the internal energy from Equation 2.12 is reformulated into:

$$\delta \boldsymbol{W} = \boldsymbol{u}^e : \boldsymbol{f}_{int} = \int \delta \boldsymbol{F}^T : \boldsymbol{P} \mathrm{d}\Omega_0 = \int \delta \boldsymbol{F}_{ij} \boldsymbol{P}_{ji} \mathrm{d}\Omega_0 \tag{2.20}$$

The variation of deformation gradient can be written as $\delta \boldsymbol{F} = \delta \boldsymbol{u}^{eT} \boldsymbol{B}_0$. It can also be written in the Index-Notation form as $\delta \boldsymbol{F}_{ij} = \delta \boldsymbol{u}^e_{ki} \tilde{\boldsymbol{B}^0_{kj}}$. Then, the internal force in matrix notation can be derived as:

$$\delta \boldsymbol{W} = \delta u^e : \boldsymbol{f}_{int} = \boldsymbol{u}^e_{ki} \boldsymbol{f}^{int}_{ki} = \int \delta \boldsymbol{F}_{ji} \boldsymbol{P}_{ij} \mathrm{d}\Omega_0 = \delta \boldsymbol{u}^e_{ki} \int \tilde{\boldsymbol{B}^0_{kj}} \boldsymbol{P}_{ji} \mathrm{d}\Omega_0 = \delta \boldsymbol{u}^e : \int \tilde{\boldsymbol{B}_0} \boldsymbol{P} \mathrm{d}\Omega_0 \tag{2.21}$$

$$\boldsymbol{f}_{int} = \int \tilde{\boldsymbol{B}_0} \boldsymbol{P} \mathrm{d}\Omega_0 \tag{2.22}$$

The tangential stiffness matrix is determined by variation of the force vector, which is determined by the velocity of nodal displacement $\dot{\boldsymbol{u}}^e$. Because Jacobian Matrix is relatively complex to build, the derivative of internal force $\dot{\boldsymbol{f}}_{int}$ in time is as follows:

$$\dot{\boldsymbol{f}_{int}} = \int \tilde{\boldsymbol{B}_0} \dot{\boldsymbol{S}} \boldsymbol{F}^T \mathrm{d}\Omega_0 + \int \tilde{\boldsymbol{B}_0} \boldsymbol{S} \dot{\boldsymbol{F}^T} \mathrm{d}\Omega_0 \tag{2.23}$$

The first term corresponds to material stiffness because the time derivative of second Piola-Kirchhoff-Stress tensor is time-dependent part. And the second term conresponds to geometrical stiffness, as is shown in equation 2.18. Because the time derivative $\dot{\boldsymbol{F}}$ equals $\dot{\boldsymbol{u}^e}^T \tilde{\boldsymbol{B}_0}$, Equation 2.23 can be transformed as:

$$\dot{\boldsymbol{f}}_{int,geo} = \int \tilde{\boldsymbol{B}_0} \boldsymbol{S} \tilde{\boldsymbol{B}_0} \Omega_0 \dot{\boldsymbol{u}}^e \tag{2.24}$$

The geometrical stiffness matrix can be achieved by coupling the temporal variation of non-linear force with the displacements using the tangential stiffness matrix.

$$\dot{\boldsymbol{f}}_{int,geo} = \boldsymbol{K}_{geo} \dot{\boldsymbol{u}}^e = \int \tilde{\boldsymbol{B}_0} \boldsymbol{S} \tilde{\boldsymbol{B}_0} \Omega_0 \dot{\boldsymbol{u}}^e \tag{2.25}$$

The last step in determining the non-linear force and the tangential stiffness matrix is the integrate the non-linear force on domain $d\Omega_0$, which is the transformation from domain $d\Omega_0$ to reference coordinate system:

$$\int f(x) \mathrm{d}\Omega = \int f(x) \frac{\partial \xi}{\partial \xi} \mathrm{d}\Omega = \int f(x) \frac{\partial \Omega}{\partial \xi} \mathrm{d}\xi = \int f(x) \, det \left(\frac{\partial X}{\partial \xi} \right) \mathrm{d}\xi \tag{2.26}$$

2.2.1 Gauss Integration

Numerical integration plays an important role in FEM. Gauss integration is an efficient approach compared to other numerical integration approaches. It integrates a function $\boldsymbol{f}(\boldsymbol{\xi})$ on the spatial domain Ω by replacing a summation of certain function values at the so-called Gauss points $\tilde{\boldsymbol{\xi}}$ that are each multiplied by a scalar (weight) $\boldsymbol{\omega}$. According to the work of W. A. Wall [Wall 2014 p.73], a clear expression of Gauss integration is:

$$\int_E f(\xi)\,\mathrm{d}\xi_1\mathrm{d}\xi_2\mathrm{d}\xi_3 = \sum_{i=1}^{numgp_1}\sum_{j=1}^{numgp_2}\sum_{k=1}^{numgp_3} f\left(\tilde{\xi}_1^i\tilde{\xi}_2^j\tilde{\xi}_3^k\right)\cdot w_1^i w_2^j w_3^k \tag{2.27}$$

In this expression, the corresponding weights are applied for each of the three spatial directions, which is available to tak Gauss points coordinates and weights tabulated for the one-dimensional case. The locations and weights of these points can be expressed theoretically in Table 2.1.

2.3 Stress Recovery

2.3.1 Extrapolation of Stress from Gauss points

Stress recovery is an approach to achieve an approximation of nodal stress by extrapolation of stress from Gauss points. This method is regarded as advanced technique with respect to the method, which direct evaluates the stress at nodes. In this paper, we name the latter method as Direct-Evaluation-Algorithms. According to the research of Carlos A. Felippa [Felippa 2004 section 28, p.5], when referring to the Direct-Evaluation-Algorithms, the relative error are typically 10% to 20%, this is the main reason why we use the Stress-Recovery-Algorithms. An example in Figure 2.3 is good to describe the procedure of calculation. We have a quadrangle with quadratic shape function. Its four nodes are denoted as 1, 2, 3, 4. The Gauss points in this element are denoted as $1'$, $2'$, $3'$, $4'$. The position of each Gauss points are ξ_i', η_i' $(i = 1, 2, 3, 4)$. The shape functions of each Gauss points can be written in:

$$N_1'(\xi', \eta') = \frac{1}{4}\times(1-\xi')\times(1-\eta') \tag{2.28}$$

$$N_2'(\xi', \eta') = \frac{1}{4}\times(1+\xi')\times(1-\eta') \tag{2.29}$$

$$N_3'(\xi', \eta') = \frac{1}{4}\times(1+\xi')\times(1+\eta') \tag{2.30}$$

$$N_4'(\xi', \eta') = \frac{1}{4}\times(1-\xi')\times(1+\eta') \tag{2.31}$$

We assume that the stress at Gauss points have been already calculated out in previous step and they can written as σ_1', σ_2', σ_3', σ_4'. The nodal stress components are calculated by a bilinear extrapolation based on the given values at Gauss points. We replacing the coordinates of nodal

Table 2.1: Gauss-Legendre points and weights

numgp	i	$\tilde{\xi}^i$	ω^i
1	1	0	2
2	1	$-1/\sqrt{3}$	1
	2	$+1/\sqrt{3}$	1
3	1	$-\sqrt{3/5}$	5/9
	2	0	8/9
	3	$+\sqrt{3/5}$	5/9
4	1	$-\sqrt{\left(15+\sqrt{120}\right)/35}$	$\left(18-\sqrt{30}\right)/36$
	2	$-\sqrt{\left(15-\sqrt{120}\right)/35}$	$\left(18+\sqrt{30}\right)/36$
	3	$+\sqrt{\left(15-\sqrt{120}\right)/35}$	$\left(18+\sqrt{30}\right)/36$
	4	$+\sqrt{\left(15+\sqrt{120}\right)/35}$	$\left(18-\sqrt{30}\right)/36$
5	1	$-1/3\sqrt{5+2\sqrt{10/7}}$	$\left(332-13\sqrt{70}\right)/900$
	2	$-1/3\sqrt{5-2\sqrt{10/7}}$	$\left(332+13\sqrt{70}\right)/900$
	3	0	128/255
	4	$+1/3\sqrt{5-2\sqrt{10/7}}$	$\left(332+13\sqrt{70}\right)/900$
	5	$+1/3\sqrt{5+2\sqrt{10/7}}$	$\left(332-13\sqrt{70}\right)/900$

points into the shape functions of Gauss points for calculating the nodal stress, which can be derived as:

$$\sigma_1 = \sigma_1{}' N_1{}' \left(\xi_1, \eta_1\right) + \sigma_2{}' N_2{}' \left(\xi_2, \eta_2\right) + \sigma_3{}' N_3{}' \left(\xi_3, \eta_3\right) + \sigma_4{}' N_4{}' \left(\xi_4, \eta_4\right) \tag{2.32}$$

$$\sigma_2 = \sigma_1{}' N_1{}' \left(\xi_1, \eta_1\right) + \sigma_2{}' N_2{}' \left(\xi_2, \eta_2\right) + \sigma_3{}' N_3{}' \left(\xi_3, \eta_3\right) + \sigma_4{}' N_4{}' \left(\xi_4, \eta_4\right) \tag{2.33}$$

$$\sigma_3 = \sigma_1{}' N_1{}' \left(\xi_1, \eta_1\right) + \sigma_2{}' N_2{}' \left(\xi_2, \eta_2\right) + \sigma_3{}' N_3{}' \left(\xi_3, \eta_3\right) + \sigma_4{}' N_4{}' \left(\xi_4, \eta_4\right) \tag{2.34}$$

$$\sigma_4 = \sigma_1{}' N_1{}' \left(\xi_1, \eta_1\right) + \sigma_2{}' N_2{}' \left(\xi_2, \eta_2\right) + \sigma_3{}' N_3{}' \left(\xi_3, \eta_3\right) + \sigma_4{}' N_4{}' \left(\xi_4, \eta_4\right) \tag{2.35}$$

Then we can form them as:

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \end{pmatrix} = \begin{pmatrix} N_1{}' \left(\xi_1, \eta_1\right) & N_2{}' \left(\xi_2, \eta_2\right) & N_3{}' \left(\xi_3, \eta_3\right) & N_4{}' \left(\xi_4, \eta_4\right) \\ N_1{}' \left(\xi_1, \eta_1\right) & N_2{}' \left(\xi_2, \eta_2\right) & N_3{}' \left(\xi_3, \eta_3\right) & N_4{}' \left(\xi_4, \eta_4\right) \\ N_1{}' \left(\xi_1, \eta_1\right) & N_2{}' \left(\xi_2, \eta_2\right) & N_3{}' \left(\xi_3, \eta_3\right) & N_4{}' \left(\xi_4, \eta_4\right) \\ N_1{}' \left(\xi_1, \eta_1\right) & N_2{}' \left(\xi_2, \eta_4\right) & N_3{}' \left(\xi_3, \eta_3\right) & N_4{}' \left(\xi_4, \eta_4\right) \end{pmatrix} \begin{pmatrix} \sigma_1{}' \\ \sigma_2{}' \\ \sigma_3{}' \\ \sigma_4{}' \end{pmatrix} \tag{2.36}$$

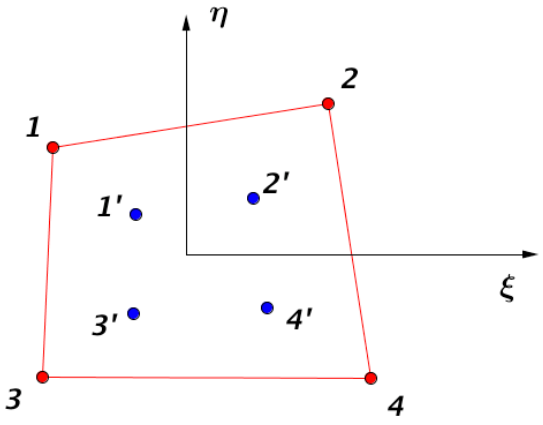

Figure 2.3: Quadrangle with four nodes

2.3.2 Quadrilateral element with four nodes: Quad4

Quad4 is used for 2D modelling of solid structure. The element can be used either as a plane element (plane stress or plane strain) or as an axisymmetric element. The element is defined by four nodes with two degrees of freedom (DOFs) at each node: translations in the nodal x and y directions. The normal Gauss integration rule for element stiffness evaluation is 2×2, as is illustrated in Figure 2.4. According to the work of Carlos A. Felippa [Felippa 2004 section 18, p.6], the components(strain, stress) are calculated at the Gauss points and are identified as k_1^G, k_2^G, k_3^G and k_4^G in Figure 2.4. Point k_i^G is closest to node k_i^E, and other Gauss point numbering follows element node numbering in the same counterclockwise manner. The natural coordinates of these points are listed in Table 2.2. The components are calculated at these Gauss points, at which strain and stress attain best approximated solutions. Then each strain and stress component is transported to the corner nodes k_1^E through k_4^E by a bilinear extrapolation based on the computed values at k_1^G through k_4^E. To understand the extrapolation procedure more clearly, we should consider the region bounded by the Gauss points as an internal Gauss element. This interpretation is depicted in Figure 2.4. The external element is denoted by (E). The internal Gauss element, denoted by (G), is also a four-node quadrilateral. The coordinate of node for element and Gauss element can be represented as k_i^G and k_i^E, respectively. The natural coordinates are denoted by ξ and η, and the internal Gauss coordinates are denoted by ξ' and η'. Both coordinates follow the simple relations from Gauss-Legendre quadrature in Table 2.2, here we have the coordinate relation of components for Quad4:

$$k_i^G = \frac{k_i^E}{\sqrt{3}}, \quad k_i^E = k_i^G \sqrt{3} \tag{2.37}$$

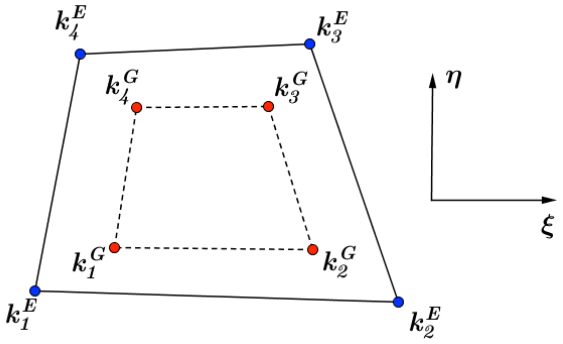

Figure 2.4: Quad4 in element coordinate and Gauss element coordinate.

The element geometry and natural coordinates are shown in Figure 2.6. There is only one type of node and associated shape function. We consider node 1 to be a typical node. From Figure 2.5 we can form:

$$N_1^e = c_1 L_{2-3} L_{3-4} \tag{2.38}$$

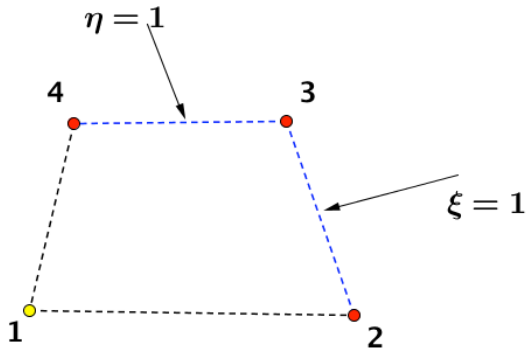

Figure 2.5: Equation of side opposite corner 1 for Quad4.

The equation of edge 2-3 is $\xi - 1 = 0$. The equation of edge 3-4 is $\eta - 1 = 0$. Inserting in Equation (2.38) results in:

$$N_1^e(\xi,\eta) = c_1(\xi - 1)(\eta - 1) = c_1(1 - \xi)(1 - \eta) \tag{2.39}$$

We evaluate the point at node 1 to find c_1 and the natural coordinates can be expressed as $\xi = \eta = -1$:

$$N_1^e(-1,-1) = c_1 \times 2 \times 2 = 4c_1 = 1 \tag{2.40}$$

Therefore, $c_1 = \frac{1}{4}$ and the shape functions is

$$N_1^e = \frac{1}{4}(1 - \xi)(1 - \eta) \tag{2.41}$$

We can use the same approach to calculate the other three nodes. Following this general expression, the shape functions of Node 2, 3 and 4 are demonstrated as:

$$N_2^e = \frac{1}{4}(1 + \xi)(1 - \eta) \tag{2.42}$$

$$N_3^e = \frac{1}{4}(1 + \xi)(1 + \eta) \tag{2.43}$$

$$N_4^e = \frac{1}{4}(1 - \xi)(1 + \eta) \tag{2.44}$$

When we have all the shape functions for Gauss element, we are able to extrapolate the components (stress, strain, etc) from Gauss points k_i^G to corner nodes k_i^E. According to Table 2.2 and Figure 2.4, we have the corner nodes in Gauss coordinate: $k_1^E(-\sqrt{3},-\sqrt{3})$, $k_2^E(\sqrt{3},-\sqrt{3})$,

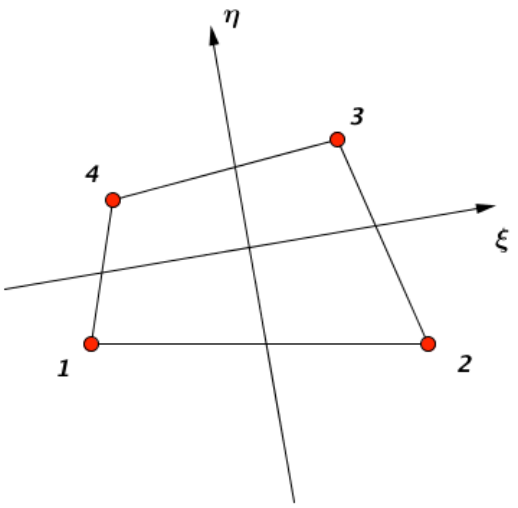

Figure 2.6: Equation of side opposite corner 1 for Quad4.

$k_3^E(\sqrt{3},\sqrt{3})$, $k_4^E(\sqrt{3},\sqrt{3})$. The extrapolation process is to replace all the coordinates of corner nodes into Gauss coordinate with the shape function of each Gauss points. Then we can get the relation between corner nodes and Gauss points as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3} \\ 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \\ w_4{}' \end{pmatrix} \tag{2.45}$$

Table 2.2: Natural Coordinate of Quad4

Corner node	ξ	η	ξ'	η'	Gauss node	ξ	η	ξ'	η'
1	-1	-1	$-\sqrt{3}$	$-\sqrt{3}$	1'	$+1/\sqrt{3}$	$-1/\sqrt{3}$	-1	-1
2	+1	-1	$+\sqrt{3}$	$-\sqrt{3}$	2'	$+1/\sqrt{3}$	$+1/\sqrt{3}$	+1	-1
3	+1	+1	$+\sqrt{3}$	$+\sqrt{3}$	3'	$+1/\sqrt{3}$	$+1/\sqrt{3}$	+1	+1
4	-1	+1	$-\sqrt{3}$	$+\sqrt{3}$	4'	$-1/\sqrt{3}$	$+1/\sqrt{3}$	-1	+1

Here w' means strains and stresses in this paper. So the w' part can be written as:

$$\begin{pmatrix} {w_1}' \\ {w_2}' \\ {w_3}' \\ {w_4}' \end{pmatrix} = \begin{pmatrix} \epsilon_{1xx} & \epsilon_{1yy} & \epsilon_{1zz} & 2\epsilon_{1yz} & 2\epsilon_{1xz} & 2\epsilon_{1xy} \\ \epsilon_{2xx} & \epsilon_{2yy} & \epsilon_{2zz} & 2\epsilon_{2yz} & 2\epsilon_{2xz} & 2\epsilon_{2xy} \\ \epsilon_{3xx} & \epsilon_{3yy} & \epsilon_{3zz} & 2\epsilon_{3yz} & 2\epsilon_{3xz} & 2\epsilon_{3xy} \\ \epsilon_{4xx} & \epsilon_{4yy} & \epsilon_{4zz} & 2\epsilon_{4yz} & 2\epsilon_{4xz} & 2\epsilon_{4xy} \end{pmatrix} \tag{2.46}$$

$$\begin{pmatrix} {w_1}' \\ {w_2}' \\ {w_3}' \\ {w_4}' \end{pmatrix} = \begin{pmatrix} \sigma_{1xx} & \sigma_{1yy} & \sigma_{1zz} & \sigma_{1yz} & \sigma_{1xz} & \sigma_{1xy} \\ \sigma_{2xx} & \sigma_{2yy} & \sigma_{2zz} & \sigma_{2yz} & \sigma_{2xz} & \sigma_{2xy} \\ \sigma_{3xx} & \sigma_{3yy} & \sigma_{3zz} & \sigma_{3yz} & \sigma_{3xz} & \sigma_{3xy} \\ \sigma_{4xx} & \sigma_{4yy} & \sigma_{4zz} & \sigma_{4yz} & \sigma_{4xz} & \sigma_{4xy} \end{pmatrix} \tag{2.47}$$

2.3.3 Quadrilateral element with eight nodes: Quad8

Quad8 is a higher order 2D, 8 node element. This element is defined by 8 nodes with two degrees of freedom (DOFs) at each node: translation in the nodal x and y direction. The element may be

used as a plane element (plane stress, plane strain) or as an axisymmetric element. For Quad8, we can choose different types of Gauss element for extrapolation. We can chose from four, eight or nine nodes Gauss element as quadrilateral elements. In this paper, we use the Gauss element with nine nodes. The nine-nodes quadrilateral Gauss element has three types of shape functions, which are associated with corner midpoint, and, center nodes, respectively. The element coordinate and Gauss element coordinate are illustrated in Figure 2.7.
According to the work of Carlos A. Felippa [Felippa 2004 section 18, p.8], the three types of shape functions are illustrated in Figure for nodes 1, 5, and 9, respectively. The procedure for calculating shape function has been clearly expressed in Section 2.3.2. Here we summarize the calculation for three typical types, represented by nodes 1, 5 and 9. The three types are illustrated in Figure 2.8.

$$N_1^e = c_1 L_{2-3} L_{3-4} L_{5-7} L_{6-8} = c_1 (\xi - 1)(\eta - 1)\xi\eta \tag{2.48}$$

$$N_5^e = c_5 L_{2-3} L_{1-4} L_{6-8} L_{3-4} = c_5 (\xi - 1)(\xi + 1)\eta(\eta - 1) = c_5 \left(1 - \xi^2\right)\eta(1 - \eta) \tag{2.49}$$

$$N_9^e = c_9 L_{1-2} L_{2-3} L_{3-4} L_{4-1} = c_9 (\xi - 1)(\eta - 1)(\xi + 1)(\eta + 1) = c_9 \left(1 - \xi^2\right)\left(1 - \eta^2\right) \tag{2.50}$$

By applying the normalization conditions, the results are:

$$c_1 = \frac{1}{4}, \quad c_5 = -\frac{1}{2}, c_9 = 1$$

By following this approach, all the shape functions can be calculated:

$$N_1 = \frac{1}{4}(\xi - 1)(\eta - 1) \cdot \xi \cdot \eta \tag{2.51}$$

$$N_2 = \frac{1}{4}(\xi + 1)(\eta - 1) \cdot \xi \cdot \eta \tag{2.52}$$

$$N_3 = \frac{1}{4}(\xi + 1)(\eta + 1) \cdot \xi \cdot \eta \tag{2.53}$$

$$N_4 = \frac{1}{4}(\xi - 1)(\eta + 1) \cdot \xi \cdot \eta \tag{2.54}$$

$$N_5 = \frac{1}{2}(1 + \xi)(1 + \xi)(\eta - 1) \cdot \eta \tag{2.55}$$

$$N_6 = \frac{1}{2}(1 + \eta)(1 - \eta)(\xi + 1) \cdot \xi \tag{2.56}$$

$$N_7 = \frac{1}{2}(1 + \xi)(1 + \xi)(\eta + 1) \cdot \eta \tag{2.57}$$

$$N_8 = \frac{1}{2}(1 + \eta)(1 - \eta)(\xi - 1) \cdot \xi \tag{2.58}$$

$$N_9 = (1 + \xi)(1 - \xi)(1 + \eta)(1 - \eta) \tag{2.59}$$

Same as in Quad4, the extrapolation function can be expressed by inserting corner nodes in Gauss coordinate into the shape function of Gauss points:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_2 & a_4 & a_5 & a_5 & a_4 & a_6 \\ a_2 & a_1 & a_2 & a_3 & a_4 & a_4 & a_5 & a_5 & a_6 \\ a_3 & a_2 & a_1 & a_2 & a_5 & a_4 & a_4 & a_5 & a_6 \\ a_2 & a_3 & a_2 & a_1 & a_5 & a_5 & a_4 & a_4 & a_6 \\ 0 & 0 & 0 & 0 & a_7 & 0 & a_8 & 0 & a_9 \\ 0 & 0 & 0 & 0 & 0 & a_7 & 0 & a_8 & a_9 \\ 0 & 0 & 0 & 0 & a_8 & 0 & a_7 & 0 & a_9 \\ 0 & 0 & 0 & 0 & 0 & a_8 & 0 & a_7 & a_9 \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \\ w_4{}' \\ w_5{}' \\ w_6{}' \\ w_7{}' \\ w_8{}' \\ w_9{}' \end{pmatrix} \tag{2.60}$$

$$\begin{aligned} a_1 &= +2.1869398 \\ a_2 &= +0.2777778 \\ a_3 &= +0.0352824 \\ a_4 &= -0.9858870 \\ a_5 &= -0.1252241 \\ a_6 &= +0.4444444 \\ a_7 &= +1.4788331 \\ a_8 &= +0.1878361 \\ a_9 &= -0.6666666 \end{aligned}$$

2.3.4 Triangular element with three nodes: Tri3

Tri3 is used for 2D modelling of solid structure. This element is defined by 3 nodes with two degrees of freedom (DOFs) at each node: translation in the nodal x and y direction. The geometry

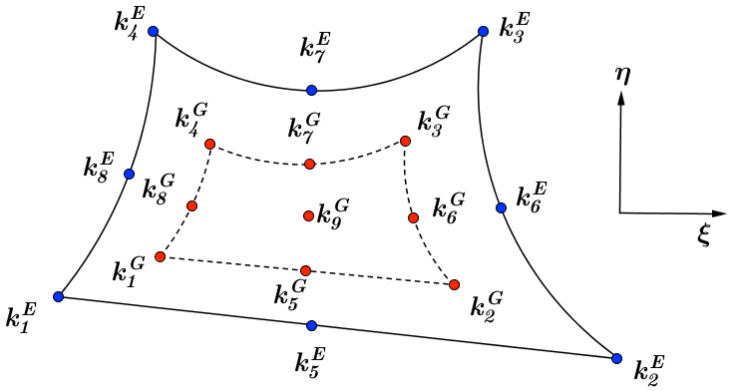

Figure 2.7: Quad8 in element coordinate and Gauss element coordinate.

of a 3-node triangle, as is shown in Figure 2.9, is specified by the location of its three corner nodes on the $\{x, y\}$ plane. The shape functions for triangular element has a different form, in comparison to quadrilateral elements. The triangular element has its own coordinates, named the triangular coordinates. According to the work of Carlos A. Felippa [Felippa 2004 section 18, p.4-5], the shape functions based on this coordinate can be formed as: $N_i = \xi_i$ for $i = 1, 2, 3$. The shape function can be derived from the following approach: the equation of the triangle edge corresponding to node i is $L_{j-k} = \xi_i = 0$, in which j and k stand for the cyclic permutations of i. See Figure 2.10 for $i = 1$, $j = 2$ and $k = 3$. Hence the general equation can be formed as:

$$N_i^e = c_i L_i \tag{2.61}$$

For finding c_1, we can evaluate $N_1^e\left(\xi_1, \xi_2, \xi_3\right)$ at node 1, which the triangular coordinates are ξ_1 =1, $\xi_2 = \xi_3 = 0$. Hence the shape function of this node can be expressed as $N_1^e(1, 0, 0) = c_1 \times 1 = 1$. Therefore $c_1 = 1$, using the same approach, $c_2 = 1$, $c_3 = 1$. Thus the shape functions are:

$$N_1^e = L_1, \quad N_2^e = L_2, \quad N_3^e = L_3 \tag{2.62}$$

By combining the shape function and corner coordinate in Gauss element from Table 2.3, the extrapolation of Tri3 can be written as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 5/3 & -1/3 & -1/3 \\ -1/3 & 5/3 & -1/3 \\ -1/3 & -1/3 & 5/3 \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \end{pmatrix} \tag{2.63}$$

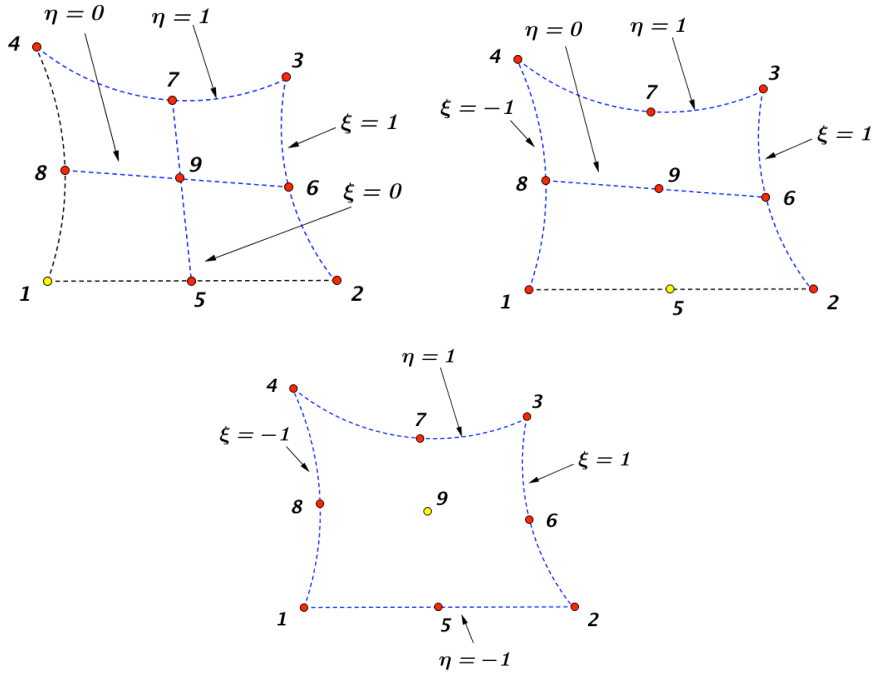

Figure 2.8: Equation of side opposite corner 1 for Quad8.

According to the work of Carlos A. Felippa [Felippa 2004 section 15 p.5-6], it is good to express the quantities in triangular coordinates when the type of element are formed by triangular. However, quantities such as displacements, strains and stresses are often calculated in the Cartesian coordinate system. Thus, the transformation functions from Cartesian coordinate system to the triangular coordinate system is necessary. The following equation shows the relation between this two systems:

$$\begin{pmatrix} 1 \\ x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_2 \\ y_1 & y_2 & y_2 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} \tag{2.64}$$

Table 2.3: Natural Coordinate of Tri3

Corner node	ξ	η	ξ'	η'	Gauss node	ξ	η	ξ'	η'
1	-1	-1	-5/3	-5/3	1'	-2/3	-2/3	-1	-1
2	+1	-1	+7/3	-5/3	2'	+1/3	-2/3	+1	-1
3	-1	+1	-5/3	+7/3	3'	-2/3	+1/3	-1	+1

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} 2A_{23} & y_{23} & x_{32} \\ 2A_{31} & y_{31} & x_{13} \\ 2A_{12} & y_{12} & x_{21} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} \tag{2.65}$$

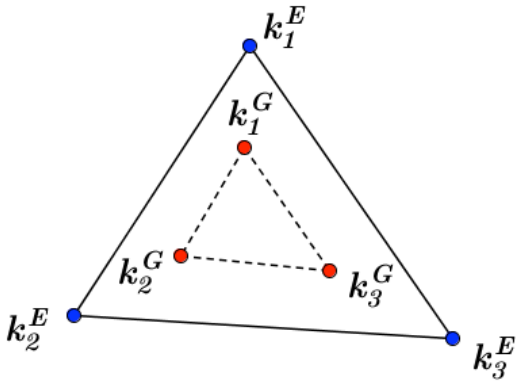

Figure 2.9: Tri3 in element coordinate and Gauss element coordinate.

2.3.5 Triangular element with six nodes: Tri6

Tri6 is a higher order 2D, 6 node element. This element is defined by 6 nodes with two degrees of freedom (DOFs) at each node: translation in the nodal x and y direction. The geometry of

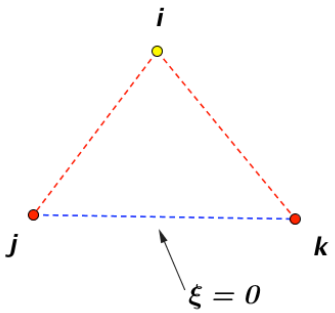

Figure 2.10: Equation of side opposite corner 1 for Tri3.

Tri6 is shown in Figure 2.11. Two kinds of Gauss elements can be applied in this case, namely, triangle with three nodes and triangle with six nodes. Here we explain the case with six nodes. According to the work of Carlos A. Felippa [Felippa 2004 section 18, p.5-6], there are two types of nodes within Tri6: corners nodes(1, 2 and 3) and midside nodes(4, 5 and 6). In both cases, we calculate the product of the two linear functions in the triangular coordinates because the shape function should be quadratic. The both types of shape function are depicted in Figure 2.12.
For corner node 1, we can derive the shape function, which refers to Figure 2.12 at left hand side.

$$N_1^e = c_1 L_{2-3} L_{4-6} \tag{2.66}$$

For calculating N_1^e will take edge 2-5-3 and edge 4-6 into consideration. This makes the function zero at node 2,3,4,5,6 and nonzero at node 1. The value of function can be adjusted to 1 if c_1 is appropriately chosen. The equations of the edges that show in Equation 2.66 are:

$$L_{2-3} : \xi_1 = 0, \quad L_{4-6} : \xi_1 - \frac{1}{2} = 0 \tag{2.67}$$

Inserting the above functions into Equation 2.66:

$$N_1^e = c_1 \xi_1 \left(\xi_1 - \frac{1}{2}\right) \tag{2.68}$$

Same as in Tri3, $N_1^e(1,0,0) = c_1 \times 1 \times \frac{1}{2} = 1$. Then the result $c_1 = 2$ can be achieved and the corresponding shape function can be expressed as:

$$N_1^e = 2\xi_1 \left(\xi_1 - \frac{1}{2}\right) = \xi_1 \left(2\xi_1 - 1\right) \tag{2.69}$$

For midside node 4, we can express the shape function by observation of Figure 2.12:

$$N_4^e = c_4 L_{2-3} L_{1-3} \tag{2.70}$$

The equations of edges L_{2-3} and L_{1-3} are $\xi_1 = 0$ and $\xi_2 = 0$, respectively. Therefore, $N_4^e\left(\xi_1, \xi_2, \xi_3\right) = c_4 \xi_1 \xi_2$. We evaluate this function at node 4 to find c_4, and its triangular coordinates are $\xi_1 = \xi_2 =$

$\frac{1}{2}, \xi_3 = 0$. Then $N_4^e\left(\frac{1}{2}, \frac{1}{2}, 0\right) = c_4 \times \frac{1}{2} \times \frac{1}{2} = 1$. Hence $c_4 = 4$ and the shape function can be formed as:

$$N_4^e = 4\xi_1\xi_2 \tag{2.71}$$

The rest of the shape functions can be solved with by the same approach. They can be expressed as:

$$N_2^e = \xi_2\left(2\xi_2 - 1\right) \tag{2.72}$$

$$N_3^e = \xi_3\left(2\xi_3 - 1\right) \tag{2.73}$$

$$N_5^e = 4\xi_2\xi_3 \tag{2.74}$$

$$N_6^e = 4\xi_1\xi_3 \tag{2.75}$$

From the above functions, the extrapolation function of Tri6 can be written as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{pmatrix} = \begin{pmatrix} a_1 & a_1 & a_2 & a_3 & a_4 & a_4 \\ a_1 & a_2 & a_1 & a_4 & a_4 & a_3 \\ a_2 & a_1 & a_1 & a_4 & a_3 & a_4 \\ a_1 & a_7 & a_7 & a_5 & a_6 & a_5 \\ a_7 & a_7 & a_1 & a_6 & a_5 & a_5 \\ a_7 & a_1 & a_7 & a_5 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \\ w_4{}' \\ w_5{}' \\ w_6{}' \end{pmatrix} \tag{2.76}$$

$a_1 = +0.55555556$
$a_2 = +3.88888889$
$a_3 = +0.44444444$
$a_4 = -2.22222222$
$a_5 = -0.88888889$
$a_6 = +1.77777778$
$a_7 = +0.22222222$

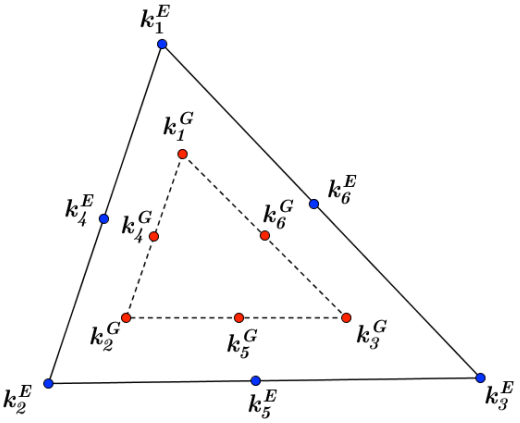

Figure 2.11: Tri6 in element coordinate and Gauss element coordinate.

2.3.6 Tetrahedral element with four nodes: Tet4

Tet4 is a higher order 3D 4-node solid element. The element is defined by 4 nodes with three degrees of freedom (DOFs) per node: translation in the nodal x, y and z directions. The geometry of Tet4 is shown in Figure 2.13. For the tetrahedron element, it is beneficial to use its own coordinate. The shape functions of Tet4 are:

$$N_1 = L_1 \tag{2.77}$$

$$N_2 = L_2 \tag{2.78}$$

$$N_3 = L_3 \tag{2.79}$$

$$N_4 = L_4 \tag{2.80}$$

We take a tetrahedron element with four nodes as Gauss element for Gauss integration. The relation between Gauss and nodal coordinate are represented in Table 2.4. From this table, we can conclude four corner nodes in Gauss coordinate: $k_1^E(j,k,k,k)$, $k_2^E(k,j,k,k)$, $k_3^E(k,k,j,k)$, $k_4^E(k,k,k,j)$ ($j = 1.927051, k = -0.309017$). The procedure to calculate the extrapolation function of Tet4 is similar as Tri3, because they both use their own coordinate system. After

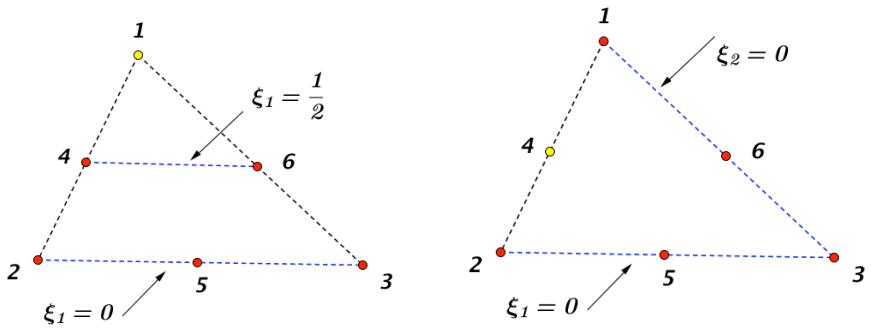

Figure 2.12: Equation of side opposite corner 1 for Tri6.

inserting the corner points into shape function, the extrapolation of Tet4 is shown as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_2 & a_2 \\ a_2 & a_1 & a_2 & a_2 \\ a_2 & a_2 & a_1 & a_2 \\ a_2 & a_2 & a_2 & a_1 \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \\ w_4{}' \end{pmatrix} \tag{2.81}$$

$$a_1 = +1.927051$$
$$a_2 = -0.309017$$

2.3.7 Tetrahedral element with ten nodes: Tet10

Tet10 is a higher order 3D 10-node solid element. The element is defined by 10 nodes with three degrees of freedom (DOFs) per node: translation in the nodal x, y and z directions. The geometry, node location, and coordinate system for this element are shown in Figure 2.14. According to the work of Carlos A. Felippa [Felippa 2004 section 10 p.21], for Tet10 it is possible to choose different types of Gauss element, which contains 1, 4,6,14,15 and 24 integration points. We choose a tetrahedron with four nodes as internal Gauss element for this paper. The shape function of this type of Gauss element is same as the shape functions of Tet4:

$$N_1 = L_1 \tag{2.82}$$

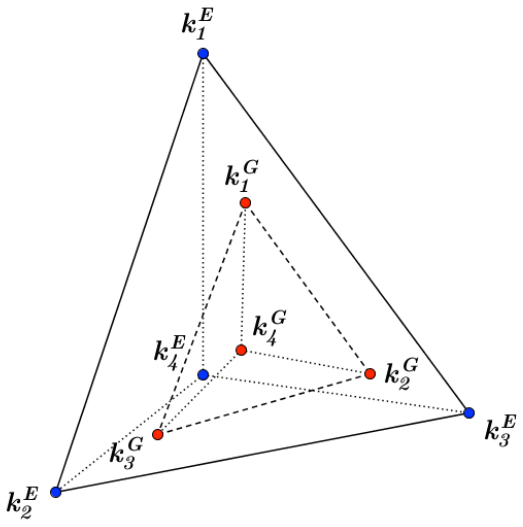

Figure 2.13: Tet4 in element coordinate and Gauss element coordinate.

$$N_2 = L_2 \tag{2.83}$$

$$N_3 = L_3 \tag{2.84}$$

$$N_4 = L_4 \tag{2.85}$$

Table 2.4: Tetrahedral Coordinate of Tet4

Corner node	L1	L2	L3	L4	L1'	L2'	L3'	L4'
1	1	0	0	0	α	β	β	β
2	0	1	0	0	β	α	β	β
3	0	0	1	0	β	β	α	β
4	0	0	0	1	β	β	β	α
Gauss node	**L1**	**L2**	**L3**	**L4**	**L1'**	**L2'**	**L3'**	**L4'**
1'	α	β	β	β	1	0	0	0
2'	β	α	β	β	0	1	0	0
3'	β	β	α	β	0	0	0	1
4'	β	β	β	α	0	0	0	1

$\alpha = 0.58541020$; $\beta = 0.13819660$

The relation between Gauss and natural coordinate of Tet10 has a similar basic frame as Tet4. The new appended six corner nodes are picked as midpoint of each edges. Table 2.5 shows the details of Gauss and nodal coordinate system, respectively. Then the extrapolation of Tet10 can be formed as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_2 & a_2 \\ a_2 & a_1 & a_2 & a_2 \\ a_2 & a_2 & a_1 & a_2 \\ a_2 & a_2 & a_2 & a_1 \\ a_3 & a_3 & a_2 & a_2 \\ a_3 & a_2 & a_3 & a_2 \\ a_3 & a_2 & a_2 & a_3 \\ a_2 & a_3 & a_3 & a_2 \\ a_2 & a_3 & a_2 & a_3 \\ a_2 & a_2 & a_3 & a_3 \end{pmatrix} \begin{pmatrix} w_1{}' \\ w_2{}' \\ w_3{}' \\ w_4{}' \end{pmatrix} \tag{2.86}$$

$$a_1 = +1.927051$$
$$a_2 = -0.309017$$
$$a_3 = +0.809017$$

2.4 Assembly

In the previous chapter, we have solved the value of strain and stress in one element. The next step is to assemble strain and stress at each node from the local view to the global view. Meanwhile,

Table 2.5: Coordinate system of Tet10

Corner node	L1	L2	L3	L4	L1'	L2'	L3'	L4'
1	1	0	0	0	α	β	β	β
2	0	1	0	0	β	α	β	β
3	0	0	1	0	β	β	α	β
4	0	0	0	1	β	β	β	α
5	0.5	0.5	0	0	$\frac{(\alpha+\beta)}{2}$	$\frac{(\alpha+\beta)}{2}$	β	β
6	0.5	0	0.5	0	$\frac{(\alpha+\beta)}{2}$	β	$\frac{(\alpha+\beta)}{2}$	β
7	0.5	0	1	0.5	$\frac{(\alpha+\beta)}{2}$	β	β	$\frac{(\alpha+\beta)}{2}$
8	0	0.5	0.5	0	β	$\frac{(\alpha+\beta)}{2}$	$\frac{(\alpha+\beta)}{2}$	β
9	0	0.5	0	0.5	β	$\frac{(\alpha+\beta)}{2}$	β	$\frac{(\alpha+\beta)}{2}$
10	0	0	0.5	0.5	β	β	$\frac{(\alpha+\beta)}{2}$	$\frac{(\alpha+\beta)}{2}$
Gauss node	L1	L2	L3	L4	L1'	L2'	L3'	L4'
1'	α	β	β	β	1	0	0	0
2'	β	α	β	β	0	1	0	0
3'	β	β	α	β	0	0	0	1
4'	β	β	β	α	0	0	0	1

$\alpha = 0.58541020$; $\beta = 0.13819660$

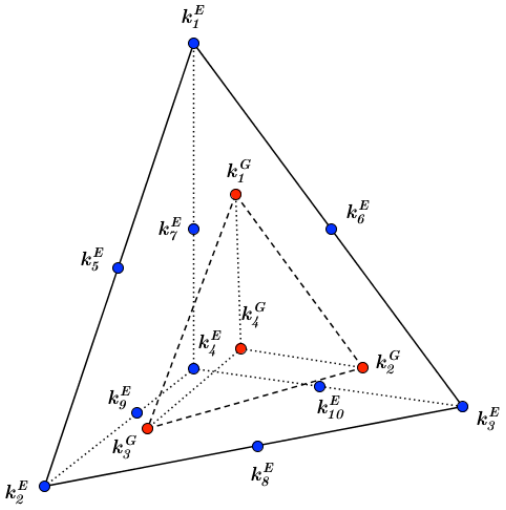

Figure 2.14: Tet10 in element coordinate and Gauss element coordinate.

element stiffness matrix and force vector are also assembled at each degrees of freedom (DOFs). According the work of W. A. Wall [Wall 2014 p.62-64], in FEM, the whole primary field can be seen as a continuous field, in which two neighboring element share one or more nodes. Now we take a simple 2D an object as an example, which is meshed by two Tri3 element as depicted in Figure 2.15. We assume that the stiffness matrix k and force vector f for each single element have been calculated beforehand. The stiffness matrix of elements is given as:

$$k^{(1)} = \begin{pmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{33}^{(1)} & k_{34}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{43}^{(1)} & k_{44}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} \\ k_{51}^{(1)} & k_{52}^{(1)} & k_{53}^{(1)} & k_{54}^{(1)} & k_{55}^{(1)} & k_{56}^{(1)} \\ k_{61}^{(1)} & k_{62}^{(1)} & k_{63}^{(1)} & k_{64}^{(1)} & k_{65}^{(1)} & k_{66}^{(1)} \end{pmatrix} \tag{2.87}$$

$$k^{(2)} = \begin{pmatrix} k_{11}^{(2)} & k_{12}^{(2)} & k_{13}^{(2)} & k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\ k_{31}^{(2)} & k_{32}^{(2)} & k_{33}^{(2)} & k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\ k_{41}^{(2)} & k_{42}^{(2)} & k_{43}^{(2)} & k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\ k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\ k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)} \end{pmatrix} \tag{2.88}$$

Now we can assemble both stiffness matrix contribution, namely from Equation 2.87 and from Equation 2.88, at the same degree of freedom. The procedure of assembly is represented as follows:

$$K = \begin{pmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & 0 & 0 \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & 0 & 0 \\ k_{51}^{(1)} & k_{52}^{(1)} & k_{55}^{(1)} & k_{56}^{(1)} & k_{53}^{(1)} & k_{54}^{(1)} & 0 & 0 \\ k_{61}^{(1)} & k_{62}^{(1)} & k_{65}^{(1)} & k_{66}^{(1)} & k_{63}^{(1)} & k_{64}^{(1)} & 0 & 0 \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} & k_{33}^{(1)} & k_{34}^{(1)} & 0 & 0 \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} & k_{43}^{(1)} & k_{44}^{(1)} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{11}^{(2)} & k_{12}^{(2)} & k_{13}^{(2)} & k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ 0 & 0 & k_{21}^{(2)} & k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\ 0 & 0 & k_{31}^{(2)} & k_{32}^{(2)} & k_{33}^{(2)} & k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\ 0 & 0 & k_{41}^{(2)} & k_{42}^{(2)} & k_{43}^{(2)} & k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\ 0 & 0 & k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\ 0 & 0 & k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)} \end{pmatrix}$$

$$
= \begin{pmatrix}
k_{11}^{(1)} & k_{12}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & 0 & 0 \\
k_{21}^{(1)} & k_{22}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & 0 & 0 \\
k_{51}^{(1)} & k_{52}^{(1)} & k_{55}^{(1)} + k_{11}^{(2)} & k_{56}^{(1)} + k_{12}^{(2)} & k_{53}^{(1)} + k_{13}^{(2)} & k_{54}^{(1)} + k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\
k_{61}^{(1)} & k_{62}^{(1)} & k_{65}^{(1)} + k_{21}^{(2)} & k_{66}^{(1)} + k_{22}^{(2)} & k_{63}^{(1)} + k_{23}^{(2)} & k_{64}^{(1)} + k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\
k_{31}^{(1)} & k_{32}^{(1)} & k_{35}^{(1)} + k_{31}^{(2)} & k_{36}^{(1)} + k_{32}^{(2)} & k_{33}^{(1)} + k_{33}^{(2)} & k_{34}^{(1)} + k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\
k_{41}^{(1)} & k_{42}^{(1)} & k_{45}^{(1)} + k_{41}^{(2)} & k_{46}^{(1)} + k_{42}^{(2)} & k_{43}^{(1)} + k_{43}^{(2)} & k_{44}^{(1)} + k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\
0 & 0 & k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\
0 & 0 & k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)}
\end{pmatrix} \tag{2.89}
$$

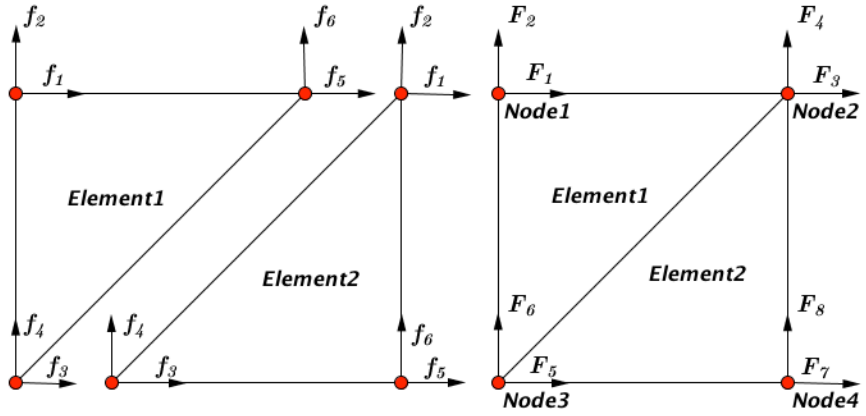

Figure 2.15: Two Tri6 elements in global and local perspective.

The same approach is also applied to force vector, because it also share the contribution at

each degree of freedom. The procedure of assembly are expressed as:

$$F = \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_5^{(1)} \\ f_6^{(1)} \\ f_3^{(1)} \\ f_4^{(1)} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \\ f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \end{pmatrix} = \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_5^{(1)} + f_1^{(2)} \\ f_6^{(1)} + f_2^{(2)} \\ f_3^{(1)} + f_3^{(2)} \\ f_4^{(1)} + f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \end{pmatrix} \tag{2.90}$$

For strain and stress, we need to assemble the contributions at each node. An extra procedure after assembling of strain or stress is solving the average value by dividing the number of elements that share the same node. Because the value of strain and stress are discontinuous between neighboring elements. The rule for averaging strain or stress can be formed as:

$$\epsilon = \sum_{i=1}^{n} \frac{\epsilon_{i,raw}}{n} \qquad \sigma = \sum_{i=1}^{n} \frac{\sigma_{i,raw}}{n} \tag{2.91}$$

$$\epsilon_{raw} = \begin{pmatrix} \epsilon^{(1)}_{1xx} & \epsilon^{(1)}_{1yy} & \epsilon^{(1)}_{1zz} & 2\epsilon^{(1)}_{1yz} & 2\epsilon^{(1)}_{1xz} & 2\epsilon^{(1)}_{1xy} \\ \epsilon^{(1)}_{2xx} & \epsilon^{(1)}_{2yy} & \epsilon^{(1)}_{2zz} & 2\epsilon^{(1)}_{2yz} & 2\epsilon^{(1)}_{2xz} & 2\epsilon^{(1)}_{2xy} \\ \epsilon^{(1)}_{5xx} & \epsilon^{(1)}_{5yy} & \epsilon^{(1)}_{5zz} & 2\epsilon^{(1)}_{5yz} & 2\epsilon^{(1)}_{5xz} & 2\epsilon^{(1)}_{5xy} \\ \epsilon^{(1)}_{6xx} & \epsilon^{(1)}_{6yy} & \epsilon^{(1)}_{6zz} & 2\epsilon^{(1)}_{6yz} & 2\epsilon^{(1)}_{6xz} & 2\epsilon^{(1)}_{6xy} \\ \epsilon^{(1)}_{3xx} & \epsilon^{(1)}_{3yy} & \epsilon^{(1)}_{3zz} & 2\epsilon^{(1)}_{3yz} & 2\epsilon^{(1)}_{3xz} & 2\epsilon^{(1)}_{3xy} \\ \epsilon^{(1)}_{4xx} & \epsilon^{(1)}_{4yy} & \epsilon^{(1)}_{4zz} & 2\epsilon^{(1)}_{4yz} & 2\epsilon^{(1)}_{4xz} & 2\epsilon^{(1)}_{4xy} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \epsilon^{(2)}_{1xx} & \epsilon^{(2)}_{1yy} & \epsilon^{(2)}_{1zz} & 2\epsilon^{(2)}_{1yz} & 2\epsilon^{(2)}_{1xz} & 2\epsilon^{(2)}_{1xy} \\ \epsilon^{(2)}_{2xx} & \epsilon^{(2)}_{2yy} & \epsilon^{(2)}_{2zz} & 2\epsilon^{(2)}_{2yz} & 2\epsilon^{(2)}_{2xz} & 2\epsilon^{(2)}_{2xy} \\ \epsilon^{(2)}_{3xx} & \epsilon^{(2)}_{3yy} & \epsilon^{(2)}_{3zz} & 2\epsilon^{(2)}_{3yz} & 2\epsilon^{(2)}_{3xz} & 2\epsilon^{(2)}_{3xy} \\ \epsilon^{(2)}_{4xx} & \epsilon^{(2)}_{4yy} & \epsilon^{(2)}_{4zz} & 2\epsilon^{(2)}_{4yz} & 2\epsilon^{(2)}_{4xz} & 2\epsilon^{(2)}_{4xy} \\ \epsilon^{(2)}_{5xx} & \epsilon^{(2)}_{5yy} & \epsilon^{(2)}_{5zz} & 2\epsilon^{(2)}_{5yz} & 2\epsilon^{(2)}_{5xz} & 2\epsilon^{(2)}_{5xy} \\ \epsilon^{(2)}_{6xx} & \epsilon^{(2)}_{6yy} & \epsilon^{(2)}_{6zz} & 2\epsilon^{(2)}_{6yz} & 2\epsilon^{(2)}_{6xz} & 2\epsilon^{(2)}_{6xy} \end{pmatrix}$$

$$\epsilon = \begin{pmatrix} \epsilon^{(1)}_{1xx} & \epsilon^{(1)}_{1yy} & \epsilon^{(1)}_{1zz} & 2\epsilon^{(1)}_{1yz} & 2\epsilon^{(1)}_{1xz} & 2\epsilon^{(1)}_{1xy} \\ \epsilon^{(1)}_{2xx} & \epsilon^{(1)}_{2yy} & \epsilon^{(1)}_{2zz} & 2\epsilon^{(1)}_{2yz} & 2\epsilon^{(1)}_{2xz} & 2\epsilon^{(1)}_{2xy} \\ \frac{\epsilon^{(1)}_{5xx}+\epsilon^{(2)}_{1xx}}{2} & \frac{\epsilon^{(1)}_{5yy}+\epsilon^{(2)}_{1yy}}{2} & \frac{\epsilon^{(1)}_{5zz}+\epsilon^{(2)}_{1zz}}{2} & \frac{2\epsilon^{(1)}_{5yz}+2\epsilon^{(2)}_{1yz}}{2} & \frac{2\epsilon^{(1)}_{5xz}+2\epsilon^{(2)}_{1xz}}{2} & \frac{2\epsilon^{(1)}_{5xy}+2\epsilon^{(2)}_{1xy}}{2} \\ \frac{\epsilon^{(1)}_{6xx}+\epsilon^{(2)}_{2xx}}{2} & \frac{\epsilon^{(1)}_{6yy}+\epsilon^{(2)}_{2yy}}{2} & \frac{\epsilon^{(1)}_{6zz}+\epsilon^{(2)}_{2zz}}{2} & \frac{2\epsilon^{(1)}_{6yz}+2\epsilon^{(2)}_{2yz}}{2} & \frac{2\epsilon^{(1)}_{6xz}+2\epsilon^{(2)}_{2xz}}{2} & \frac{2\epsilon^{(1)}_{6xy}+2\epsilon^{(2)}_{2xy}}{2} \\ \frac{\epsilon^{(1)}_{3xx}+\epsilon^{(2)}_{3xx}}{2} & \frac{\epsilon^{(1)}_{3yy}+\epsilon^{(2)}_{3yy}}{2} & \frac{\epsilon^{(1)}_{3zz}+\epsilon^{(2)}_{3zz}}{2} & \frac{2\epsilon^{(1)}_{3yz}+2\epsilon^{(2)}_{3yz}}{2} & \frac{2\epsilon^{(1)}_{3xz}+2\epsilon^{(2)}_{3xz}}{2} & \frac{2\epsilon^{(1)}_{3xy}+2\epsilon^{(2)}_{3xy}}{2} \\ \frac{\epsilon^{(1)}_{4xx}+\epsilon^{(2)}_{4xx}}{2} & \frac{\epsilon^{(1)}_{4yy}+\epsilon^{(2)}_{4yy}}{2} & \frac{\epsilon^{(1)}_{4zz}+\epsilon^{(2)}_{4zz}}{2} & \frac{2\epsilon^{(1)}_{4yz}+2\epsilon^{(2)}_{4yz}}{2} & \frac{2\epsilon^{(1)}_{4xz}+2\epsilon^{(2)}_{4xz}}{2} & \frac{2\epsilon^{(1)}_{4xy}+2\epsilon^{(2)}_{4xy}}{2} \\ \epsilon^{(2)}_{5xx} & \epsilon^{(2)}_{5yy} & \epsilon^{(2)}_{5zz} & 2\epsilon^{(2)}_{5yz} & 2\epsilon^{(2)}_{5xz} & 2\epsilon^{(2)}_{5xy} \\ \epsilon^{(2)}_{6xx} & \epsilon^{(2)}_{6yy} & \epsilon^{(2)}_{6zz} & 2\epsilon^{(2)}_{6yz} & 2\epsilon^{(2)}_{6xz} & 2\epsilon^{(2)}_{6xy} \end{pmatrix} \tag{2.92}$$

$$\sigma_{raw} = \begin{pmatrix} \sigma^{(1)}_{1xx} & \sigma^{(1)}_{1yy} & \sigma^{(1)}_{1zz} & \sigma^{(1)}_{1yz} & \sigma^{(1)}_{1xz} & \sigma^{(1)}_{1xy} \\ \sigma^{(1)}_{2xx} & \sigma^{(1)}_{2yy} & \sigma^{(1)}_{2zz} & \sigma^{(1)}_{2yz} & \sigma^{(1)}_{2xz} & f^{(1)}_{2xy} \\ \sigma^{(1)}_{5xx} & \sigma^{(1)}_{5yy} & \sigma^{(1)}_{5zz} & \sigma^{(1)}_{5yz} & \sigma^{(1)}_{5xz} & \sigma^{(1)}_{5xy} \\ \sigma^{(1)}_{6xx} & \sigma^{(1)}_{6yy} & \sigma^{(1)}_{6zz} & \sigma^{(1)}_{6yz} & \sigma^{(1)}_{6xz} & \sigma^{(1)}_{6xy} \\ \sigma^{(1)}_{3xx} & \sigma^{(1)}_{3yy} & \sigma^{(1)}_{3zz} & \sigma^{(1)}_{3yz} & \sigma^{(1)}_{3xz} & \sigma^{(1)}_{3xy} \\ \sigma^{(1)}_{4xx} & \sigma^{(1)}_{4yy} & \sigma^{(1)}_{4zz} & \sigma^{(1)}_{4yz} & \sigma^{(1)}_{4xz} & \sigma^{(1)}_{4xy} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma^{(2)}_{1xx} & \sigma^{(2)}_{1yy} & \sigma^{(2)}_{1zz} & \sigma^{(2)}_{1yz} & \sigma^{(2)}_{1xz} & \sigma^{(2)}_{1xy} \\ \sigma^{(2)}_{2xx} & \sigma^{(2)}_{2yy} & \sigma^{(2)}_{2zz} & \sigma^{(2)}_{2yz} & \sigma^{(2)}_{2xz} & \sigma^{(2)}_{2xy} \\ \sigma^{(2)}_{3xx} & \sigma^{(2)}_{3yy} & \sigma^{(2)}_{3zz} & \sigma^{(2)}_{3yz} & \sigma^{(2)}_{3xz} & \sigma^{(2)}_{3xy} \\ \sigma^{(2)}_{4xx} & \sigma^{(2)}_{4yy} & \sigma^{(2)}_{4zz} & \sigma^{(2)}_{4yz} & \sigma^{(2)}_{4xz} & \sigma^{(2)}_{4xy} \\ \sigma^{(2)}_{5xx} & \sigma^{(2)}_{5yy} & \sigma^{(2)}_{5zz} & \sigma^{(2)}_{5yz} & \sigma^{(2)}_{5xz} & \sigma^{(2)}_{5xy} \\ \sigma^{(2)}_{6xx} & \sigma^{(2)}_{6yy} & \sigma^{(2)}_{6zz} & \sigma^{(2)}_{6yz} & \sigma^{(2)}_{6xz} & \sigma^{(2)}_{6xy} \end{pmatrix}$$

$$\sigma = \begin{pmatrix} \sigma^{(1)}_{1xx} & \sigma^{(1)}_{1yy} & \sigma^{(1)}_{1zz} & \sigma^{(1)}_{1yz} & \sigma^{(1)}_{1xz} & \sigma^{(1)}_{1xy} \\ \sigma^{(1)}_{2xx} & \sigma^{(1)}_{2yy} & \sigma^{(1)}_{2zz} & \sigma^{(1)}_{2yz} & \sigma^{(1)}_{2xz} & f^{(1)}_{2xy} \\ \frac{\sigma^{(1)}_{5xx}+\sigma^{(2)}_{1xx}}{2} & \frac{\sigma^{(1)}_{5yy}+\sigma^{(2)}_{1yy}}{2} & \frac{\sigma^{(1)}_{5zz}+\sigma^{(2)}_{1zz}}{2} & \frac{\sigma^{(1)}_{5yz}+\sigma^{(2)}_{1yz}}{2} & \frac{\sigma^{(1)}_{5xz}+\sigma^{(2)}_{1xz}}{2} & \frac{\sigma^{(1)}_{5xy}+\sigma^{(2)}_{1xy}}{2} \\ \frac{\sigma^{(1)}_{6xx}+\sigma^{(2)}_{2xx}}{2} & \frac{\sigma^{(1)}_{6yy}+\sigma^{(2)}_{2yy}}{2} & \frac{\sigma^{(1)}_{6zz}+\sigma^{(2)}_{2zz}}{2} & \frac{\sigma^{(1)}_{6yz}+\sigma^{(2)}_{2yz}}{2} & \frac{\sigma^{(1)}_{6xz}+\sigma^{(2)}_{2xz}}{2} & \frac{\sigma^{(1)}_{6xy}+\sigma^{(2)}_{2xy}}{2} \\ \frac{\sigma^{(1)}_{3xx}+\sigma^{(2)}_{3xx}}{2} & \frac{\sigma^{(1)}_{3yy}+\sigma^{(2)}_{3yy}}{2} & \frac{\sigma^{(1)}_{3zz}+\sigma^{(2)}_{3zz}}{2} & \frac{\sigma^{(1)}_{3yz}+\sigma^{(2)}_{3yz}}{2} & \frac{\sigma^{(1)}_{3xz}+\sigma^{(2)}_{3xz}}{2} & \frac{\sigma^{(1)}_{3xy}+\sigma^{(2)}_{3xy}}{2} \\ \frac{\sigma^{(1)}_{4xx}+\sigma^{(2)}_{4xx}}{2} & \frac{\sigma^{(1)}_{4yy}+\sigma^{(2)}_{4yy}}{2} & \frac{\sigma^{(1)}_{4zz}+\sigma^{(2)}_{4zz}}{2} & \frac{\sigma^{(1)}_{4yz}+\sigma^{(2)}_{4yz}}{2} & \frac{\sigma^{(1)}_{4xz}+\sigma^{(2)}_{4xz}}{2} & \frac{\sigma^{(1)}_{4xy}+\sigma^{(2)}_{4xy}}{2} \\ \sigma^{(2)}_{5xx} & \sigma^{(2)}_{5yy} & \sigma^{(2)}_{5zz} & \sigma^{(2)}_{5yz} & \sigma^{(2)}_{5xz} & \sigma^{(2)}_{5xy} \\ \sigma^{(2)}_{6xx} & \sigma^{(2)}_{6yy} & \sigma^{(2)}_{6zz} & \sigma^{(2)}_{6yz} & \sigma^{(2)}_{6xz} & \sigma^{(2)}_{6xy} \end{pmatrix} \tag{2.93}$$

Chapter 3

Programming Implementation

3.1 Pre-Processing

The purpose of Pre-Processing is to build a numerical model and export data of a meshed geometry as two lists: node list and element list. The two lists are ultimately imported into AMfe Toolbox. ANSYS Parametric Design Language (APDL) code is used to create a meshed geometry in ANSYS. A simple meshed geometry is shown in Figure 3.1. The steps for building model contains a series of related path. An example of such path is:

Geometry ⇒ Element Type ⇒ Material ⇒ Mesh ⇒ Boundary Condition

After building the model, numerical data can be exported as a node list and an element list. The node list contains coordinates of nodes. The element list consists of the attributes of each element and the index of each node that belongs to the element. The element attributes are assigned to the different parts of model by referring to the appropriate entries in the element list. The pointers are simply a set of reference numbers that include the element index (ELEM), material number (MAT), element type number (TYP), real constant set number (REAL), a coordinate system number (ESY) ,and section ID number (SEC) [ANSYS 2011].. According to the example in Figure 3.1, both lists are depicted in Table 3.1 and Table 3.2. The numerical information of both lists are captured using regular expression and stored in AMfe Toolbox. For different kinds of data it is necessary to store in its specific data type. The coordinate of each node and the index of nodes from elements are stored as NumPy array. NumPy is the fundamental package for scientific computing using Python. The advantage of NumPy for storage of node and element information depends on its efficient multi-dimensional container of generic data. According to the documentation of Numpy community [Numpy 2016], "NumPy contains a powerful N-dimensional array object and sophisticated (broadcasting) functions." Moreover, it is possible to integrate Fortran code. General attributes of the element can be stored as an efficient form- Pandas DataFrames. According to the documentation of Wes Mckinney and PyData Development Team [Mckinney 2016], "Pandas is a powerful data analysis toolkit, which provides fast, flexible, and expressive data structures designed to make working with relational or labeled data both easy and intuitive. Pandas DataFrames is a two-dimensional size-mutable, potentially heterogeneous tabular data

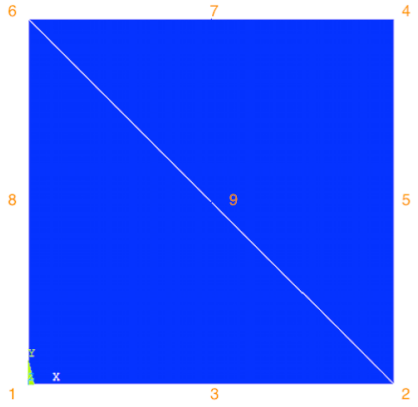

Figure 3.1: A simple meshed geometry

structure with labeled axes (rows and columns). Arithmetic operations align on both row and column labels. This format can be thought of as a dictionary-like container for Series objects."

Table 3.1: Node list exported from ANSYS

NODE	X	Y	Z	THXY	THYZ	THZX
1	0.0000	0.0000	0.0000	0.00	0.00	0.00
2	50.000	0.0000	0.0000	0.00	0.00	0.00
3	25.000	0.0000	0.0000	0.00	0.00	0.00
4	50.000	50.000	0.0000	0.00	0.00	0.00
5	50.000	25.000	0.0000	0.00	0.00	0.00
6	0.0000	50.000	0.0000	0.00	0.00	0.00
7	25.000	50.000	0.0000	0.00	0.00	0.00
8	0.0000	25.000	0.0000	0.00	0.00	0.00
9	25.000	25.000	0.0000	0.00	0.00	0.00

Table 3.2: Element list exported from ANSYS

ELEM	MAT	TYP	REL	ESY	SEC	Node	Node	Node	Node	Node	Node
1	1	1	1	0	1	6	2	4	9	5	7
2	1	1	1	0	1	6	1	2	8	3	9

Chapter 4

Element Type Test

4.1 Unit Testing with Python

According to the documentation of Fred L. Drake, Jr. [Fred L. Drake 2016], automated unit testing is the best way to determine whether the element type works well. Unit test is the batteries-included test module in the Python standard library. The motivation for writing a unit testing is to evaluate the quality of each element type and try to find the reasons for failed tests. The element types to be tested are Tri3, Tri6, Quad4, Quad8, Tet4 and Tet10. We test the strain and stress components with all the element types. As is mentioned before, the numerical solution to a mechanical problem is only an approximate solution. There is an error between the analytical solution and FEM solution for sure. Here we do not have an analytical solution to compare with our AMfe solution. The components from ANSYS will be compared to the results that are calculated from AMfe Toolbox. Although the solution from ANSYS is not the analytical solution, it could still help us check the quality of element types in AMfe Toolbox. As a comparison, we need to define both the absolute error and the relative error in the test. According to the work of Rao V. Dukkipati [Dukkipati 2010], absolute error is the magnitude of the difference between actual value and approximate value. It can be defined as:

$$e_{abs} = \|X_{actual} - X_{fem}\| \tag{4.1}$$

Relative error is the absolute error relative to the actual value. It can be defined as:

$$e_{rel} = \frac{\|X_{actual} - X_{fem}\|}{X_{actual}} \tag{4.2}$$

As Rao V. Dukkipati [Dukkipati 2010] has noted, "it is noteworthy that errors of approximation or rounding and truncation are introduced by numerical methods or algorithms and calculating with finite precision. Truncation error represents the error that occurs when some series is truncated to a fewer number of terms. Such error is virtually an algorithmic error. Roundoff error occurs because of the computing tool's inability to handle certain numbers." It is hard to set one tolerance standard for all engineering problems. The tolerance for error can vary in different situations. One thing we can do here is set the tolerance with different values and compare the results of testing between elements. In the engineering field, Von Mises stress is usually regarded as a

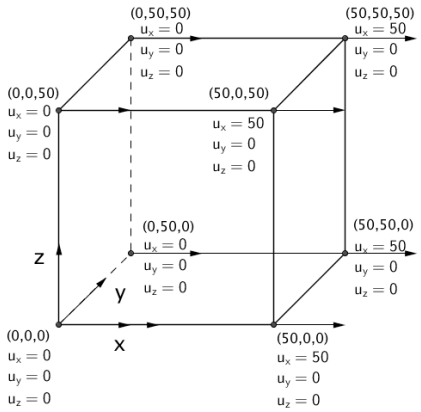

Figure 4.1: A model for test of 3D elements

failure criterion to judge a design. Hence, here we calculate the von Mises stress by meshing with each element type in their corresponding 2D or 3D models. According to the research of P. M. Kurowski [Kurowski 2012], the Von Mises stress can be expressed by the principle stresses as:

$$\sigma_{vm} = \sqrt{\frac{1}{2}\left[(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2\right]} \tag{4.3}$$

The 2D and 3D model for unit testing are build in a simple way. We create a rectangle for 2D elements and a cube for 3D elements. We apply only Dirichlet boundary condition on these models. Figure 4.2 and Figure 4.1 illustrate the model for 2D and 3D element, respectively. The complete test of all the element types is shown in Table 4.1 and Table 4.2. It is obvious that both algorithms perform well in Tri3 and Quad4. The other element types fail the unit test. It is necessary to investigate into what is wrong with the other element types.

4.2 The Patch Test

After testing the quality of each element type, we find out that the element types Tri6, Quad8, Tet4, and Tet10 are problematic and need to fixed. In order to fix these problems, we need to know if the finite element program has the correct algorithmic procedure. This is generally not easy to verify when the modelling is too complex. The patch test is a useful technique in checking the performance of element type that is being tested. The basic idea of the patch test is to apply special boundary conditions so as to maintain a constant strain and a constant stress field. This is to simplify a complex problem to a simple one, to which we already know the correct numerical solutions.

According to the theory of Carlos A. Felippa [Felippa 2004 section 15 p.9-11], the procedures of the displacement patch test are as follows: create a simple geometry that consists of one or several element; pick a patch and apply the displacement at exterior nodes and the prescribed

Table 4.1: Unit testing of the result using Direct-Evaluation-Algorithms

Tolerance Parameter	Tri3	Tri6	Quad4	Quad8	Tet4	Tet10
$e_{abs} = 10^{-10}; e_{rel} = 10^{-4}$	Fail	Fail	Fail	Fail	Fail	Fail
$e_{abs} = 10^{-10}; e_{rel} = 5 \times 10^{-4}$	Pass	Fail	Fail	Fail	Fail	Fail
$e_{abs} = 10^{-10}; e_{rel} = 10^{-3}$	Pass	Fail	Pass	Fail	Fail	Fail

displacement field will be set as $u_x = x, u_y = 0$. The reason the displacement field only applies to exterior nodes is that displacements are related to the background continuum. The last step is to verify whether the patch has a constant strain field. The mechanism behind the displacement patch test is setting the strain at every exterior node as 1, which is can be calculated by:

$$\frac{\mathrm{d}u}{\mathrm{d}x} = 1 \tag{4.4}$$

As we know,

$$\sum N_i = 1 \tag{4.5}$$

and,

$$\epsilon = \sum N_i \cdot \epsilon_i \tag{4.6}$$

Here we have $\epsilon_i = 0$ and $\sum N_i = 1$, hence the strain field is constant and equal 1.
Now we take Tri3 as an example in implementing the displacement patch test. We create a simple rectangle that consists of two Tri3 element in ANSYS, and the prescribed displacement field is depicted in Figure 4.2. Strain field from ANSYS can be viewed in Figure 4.3. As is expected, $\epsilon_{xx} = 1$. Thus, we have successfully built the model for a patch test. Now it is time to check whether AMfe Toolbox has the same constant strain field or not. We export the node, element, and displacement data from ANSYS and import all of them into AMfe Toolbox. After computing strain and strtess in AMfe, the contour plot of strain in ParaView is shown as the picture on right-hand side of Figure 4.3. The plot from AMfe Toolbox illustrates the same result of $\epsilon_{xx} = 1$. This proves that the displacement patch test has been passed. Other element types, namely Tri6, Quad4, Quad8, Tet4, and Tet10, each have a constant field. However, the results of the patch test using element Tri6, Quad8, Tet4, and Tet10 are still problematic because all of these elements have the value of 1.5 as the constant strain, as is demonstrate by the problematic result from Tet4 in Figure 4.4. It is necessary to investigate why all these elements have the same strain with the value of 1.5. By debugging the AMfe Code we can find that the strain values at Gauss points are all 1.5. This proves that the Stress-Recovery-Algorithms is correct. The problem exists in the calculation of the strain at Gauss points. Moreover, all of these element types except Tri3 and Quad4 have the same problem.

Table 4.2: Unit testing of the extrapolate result using Stress-Recovery-Algorithms

Tolerance Parameter	Tri3	Tri6	Quad4	Quad8	Tet4	Tet10
$e_{abs} = 10^{-10}; e_{rel} = 10^{-4}$	Fail	Fail	Fail	Fail	Fail	Fail
$e_{abs} = 10^{-10}; e_{rel} = 5 \times 10^{-4}$	Pass	Fail	Fail	Fail	Fail	Fail
$e_{abs} = 10^{-10}; e_{rel} = 10^{-3}$	Pass	Fail	Pass	Fail	Fail	Fail

4.3 Convergence Analysis

We can either decrease the element size or increase the polynomial degrees of the shape function to approach an relatively exact approximation. This process is denoted as convergence. A convergence analysis provides a function to reduce the relative error of the solution with increasing number of degrees of freedom [Wall 2014]. An example of convergence analysis is illustrated in Figure 4.5. On the one hand, because of the technical restriction, we can only export the node, element, and displacement lists from ANSYS and set all of them as input into AMfe Toolbox for solving strain and stress. As of now, we did not reverse the process and solve the displacement by using prescribed Neumann boundary condition and Dirichlet boundary condition as input. However, it is still a feasible plan for future research. On the other hand, we can analyse this problem using another software, Gmsh, as Pre-Processing tool. The procedures of our convergence analysis for investigating 2D elements are as follows: first, we create a 2D geometry in ANSYS and generate a fine meshing with element types Quad4, Quad8, Tri3, and Tri6, respectively. Then we pick one node to view the displacement. In this case, we do not have the analytical solution, thus the displacement result in this fine meshing will be regarded as a referential solution. In next step, we create an identical modelling in Gmsh and refine the meshing of all the 2D element types by decreasing the size of the elements. Then, we can record the displacement results of both AMfe and ANSYS from meshing with different degrees of freedom (DOFs), and plot the corresponding ratios of AMfe results to ANSYS results. The 3D element types, Tet4 and Tet10, can also be analysed using this approach. An example of 2D model meshing with Tri3 is shown in Figure 4.6 and an example of 3D model meshing with Tet4 is shown in Figure 4.7. The displacement for each of the 2D element types has been recorded in Table 4.3 and Table 4.4. The displacement for 3D element types are represented in Table 4.5. The same nodal displacements from a fine meshing in ANSYS are:

$$2D : Ux = -0.0015045; Uy = 0.011738$$

$$3D : Ux = 9.47; Uy = 0; Uz = 0$$

The convergence plots of 2D element types are depicted in Figure 4.8 and Figure 4.9. Figure 4.10 shows the convergence performance of 3D element types. All the element types show a

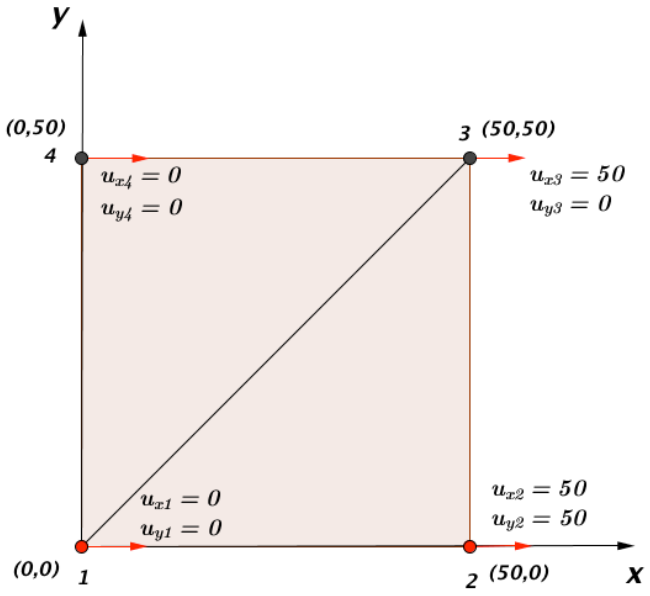

Figure 4.2: A model for test of 2D elements

good convergence to the exact solution. The higher order element types (Tri6 and Tet10) have a relatively higher convergence rate than the lower order element types (Tri3 and Tet4) do. This characteristic is not that discernible in quadrilateral element types (Quad4 and Quad8).

It is also important to check the convergence of Von Mises stress. Thus we refine the meshing in a 2D or 3D modelling using all the element types in ANSYS, and then export the data of node, element, and displacement from ANSYS and import them into AMfe Toolbox. The next step is to run the code to get the computing results for our convergence study. σ_{xy}, σ_{yy}, σ_{zz} are for 2D convergence analysis and σ_{xy}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{yz}, σ_{xz} are for 3D convergence analysis. Then we can use these values to calculate the Von Mises stress:

$$\sigma_v = \sqrt{\frac{1}{2}\left[\left(\sigma_{11}-\sigma_{22}\right)^2 + \left(\sigma_{22}-\sigma_{33}\right)^2 + \left(\sigma_{33}-\sigma_{11}\right)^2 + 6\left({\sigma_{12}}^2 + {\sigma_{23}}^2 + {\sigma_{31}}^2\right)\right]} \tag{4.7}$$

For a general plane stress case with restrictions $\sigma_3 = 0$, $\sigma_{31} = \sigma_{23} = 0$, we can simplify the Von Mises equation as:

$$\sigma_v = \sqrt{{\sigma_{11}}^2 - \sigma_{11}\sigma_{22} + {\sigma_{22}}^2 + 3{\sigma_{12}}^2} \tag{4.8}$$

The 2D model is built as shown in Figure 4.11 and the 3D model is built as shown in Figure 4.12. Table 4.6 shows the test data of convergence analysis for all element types. Figure 4.13 illustrate the relation between the DOFs of mesh and the Von Mises stress of the observer point, which is denoted C in 2D model and is denoted D in 3D model. The three plots in Figure 4.13 show that all the element types converges to the exact solution and the elements with higher order (Tri6,

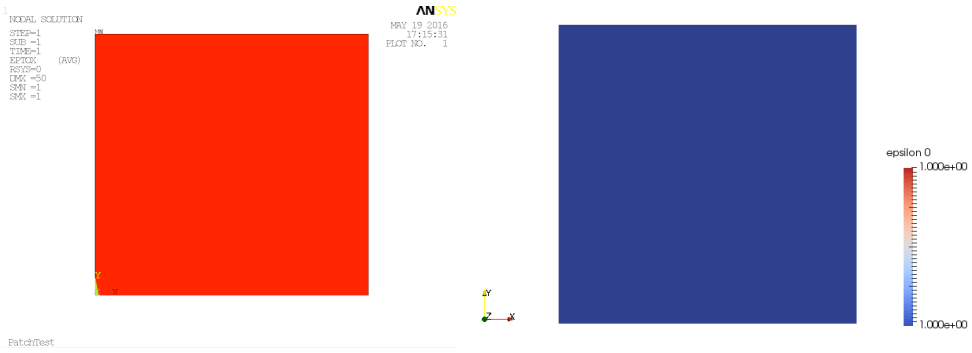

Figure 4.3: left: Contour plot of ϵ_{xx} from ANSYS (SMN: minimal value; SMX: maximal value); right: Contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

Quad8, and Tet10) converges faster than the the one with lower order (Tri3, Quad4, and Tet4). When we compare the two 2D element Tri3 and Quad4, we can find the Quad4 has a higher convergence rate than Tri3.

4.4 Complex Model Test

The patch test is only applicable to the simple models, it is meaningful how these element type behave in a complex model. Now we create a relatively complex model meshing with each element type and calculate in AMfe by both algorithm, namely, direct evaluation at nodes and extrapolation from Gauss points, namely, Stress-Recovery-Algorithms. The main objective of complex model test is to compare the strain and stress contour plot of each element type in ANSYS and in AMfe Toolbox with both algorithm. Furthermore, the comparison between the two algorithm is also significant for our research. The results from AMfe Toolbox display in ParaView as a Post-Processing tool. It is worth noting that the order of components for tensors follows this mapping rule: 0-xx; 1-yy; 2-zz; 3-yy; 4-yz; 5-xz; 6-zz; 7-xz; 8-xy. For a clear comparison and good contrast, we import a same color map for ANSYS and ParaView. However, there exists still a clear chromatic aberration between the contour plots from ANSYS and ParaView. Figure 4.9 shows an example to display the color difference. This issue may be fixed when we using an advanced ANSYS version, because ANSYS 12.0, which we use in this paper, can only offers a 7-bit color in color map.

Figure 4.15 to Figure 4.20 illustrate the contour plot of strain and stress in all direction. The contour plots from AMfe are meshed with element type Tri3 and the contour plot from ANSYS is meshed with element PLANE42(triangular option). According to the documentation of ANSYS [**APDL**], "PLANE42 is used for 2D modeling of solid structures. The element can be used either as a plane element (plane stress or plane strain) or as an axisymmetric element. The element is defined by four nodes having two degrees of freedom at each node: translations in the nodal x and y directions." From the comparison we can find that there is no difference between the both algorithm in AMfe. The results of all components are very similar to the ANSYS solution.

Figure 4.21 to Figure 4.26 illustrate the contour plot of strain and stress in all direction.

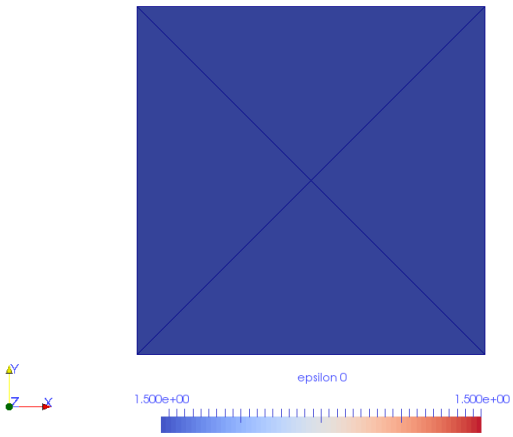

Figure 4.4: Patch Test of Tet4

The contour plots from AMfe are meshed with element type Tri6 and the contour plot from ANSYS is meshed with element PLANE2. According to the documentation of ANSYS [**APDL**], "PLANE2 is a 6-node triangular element. The element is defined by six nodes having two degrees of freedom at each node: translations in the nodal x and y directions." The plots show that the solution with Stress-Recovery-Algorithms is relative close to the ANSYS solution, the algorithm without Stress-Recovery-Algorithm has a higher strain and stress than ANSYS solution at corner points and the inner side of circle.

Figure 4.27 to Figure 4.32 illustrate the contour plot of strain and stress in all direction. The

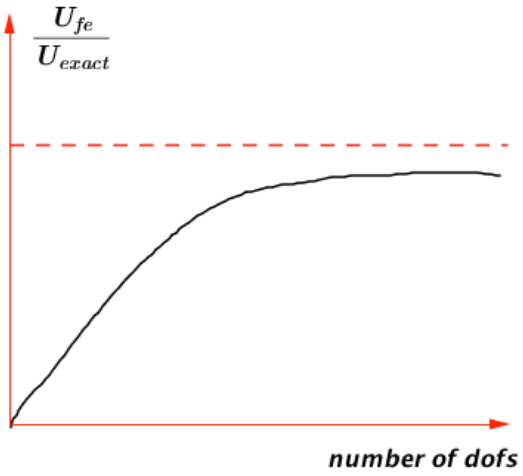

Figure 4.5: Convergence analysis

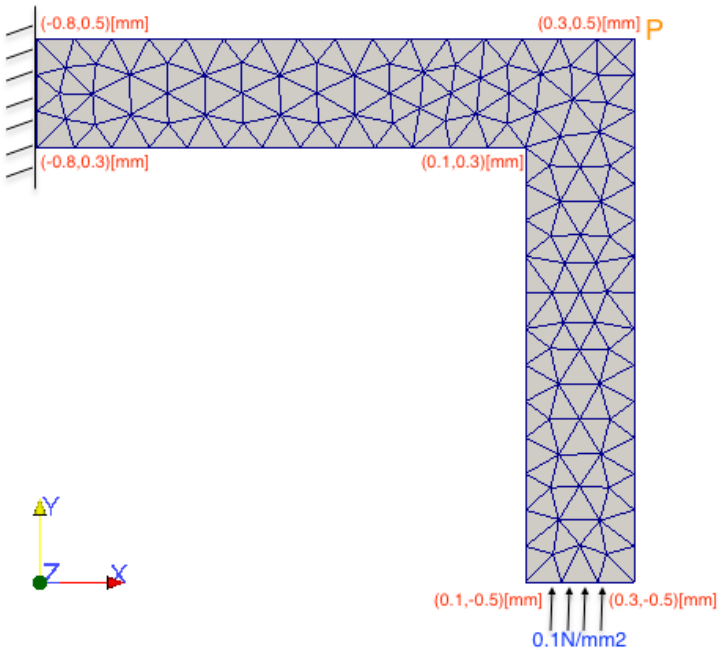

Figure 4.6: 2D model meshing with Tri3 for displacement convergence analysis; P as observer point

contour plots from AMfe are meshed with element type Quad4 and the contour plot from ANSYS is meshed with element PLANE42 (quadrangle option). We can observe from the plots that the solution by Stress-Recovery-Algorithms is close to ANSYS solution. The deviation of results of the both algorithms appear at some position as it shown in Plots with Tri6.

Figure 4.33 to Figure 4.38 illustrate the contour plot of strain and stress in all direction. The contour plots from AMfe are meshed with element type Quad8 and the contour plot from ANSYS is meshed with element PLANE82. According to the documentation of ANSYS [**APDL**], "Plane82 is a higher order version of the 2D, four node element (PLANE82). It provides more accurate results for mixed (quadrilateral-triangular) automatic meshes and can tolerate irregular shapes without as much loss of accuracy. The 8-node element is defined by eight nodes having two degrees of freedom at each node: translations in nodal x and y directions." The comparison is similar as the result, which is meshed with Quad4.

Figure 4.39 to Figure 4.48 illustrate the contour plot of strain and stress in all direction. The contour plots from AMfe are meshed with element type Tet4 and the contour plot from ANSYS is meshed with element SOLID45. According to the documentation of ANSYS [**APDL**], "SOLID45 is used for the 3D modeling of solid structures. The element is defined by eight nodes having three degrees of freedom at each node: translations in the nodal x, y, and z directions." The plots represents that there is no difference between the both algorithm in AMfe. The both solution of AMfe have a relative bad performance in shear strain/stress than normal strain/stress.

Figure 4.51 to Figure 4.60 illustrate the contour plot of strain and stress in all direction. The

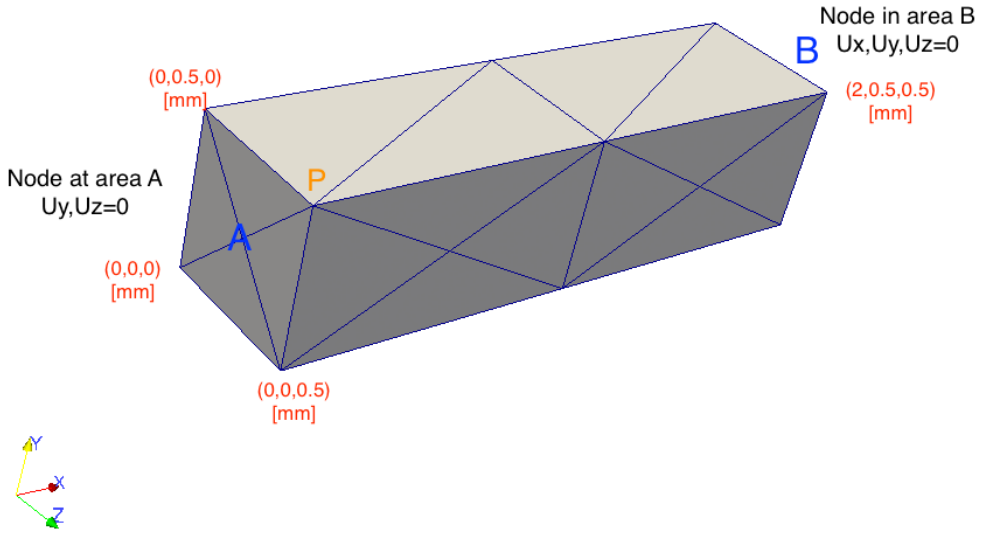

Figure 4.7: 3D model meshing with Tet4 for displacement convergence analysis; P as observer point

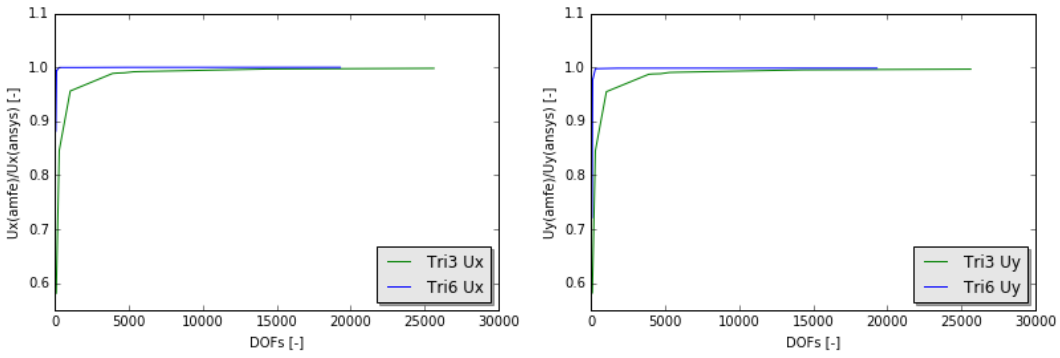

Figure 4.8: Convergence plot of Tri3 and Tri6

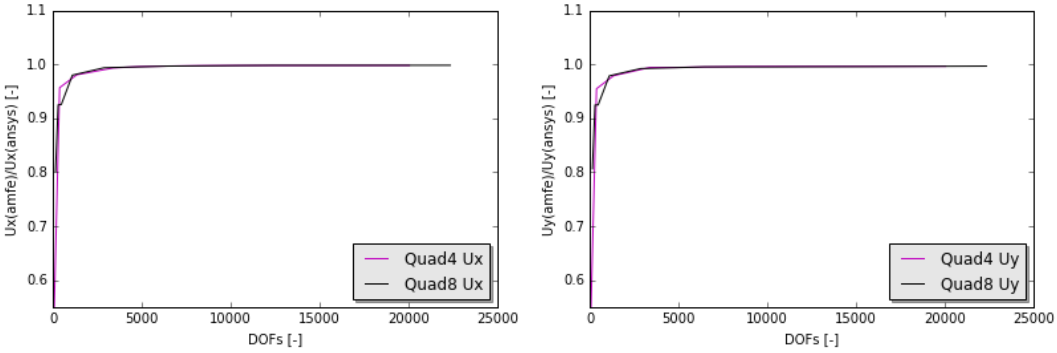

Figure 4.9: Convergence plot of Quad4 and Quad8

contour plots from AMfe are meshed with element type Tet10 and the contour plot from ANSYS is meshed with element SOLID187 (tetrahedron option). According to the documentation of ANSYS [**APDL**], "SOLID187 element is a higher order 3D, 10-node element. SOLID187 has a quadratic displacement behavior and is well suited to modeling irregular meshes. The element is defined by 10 nodes having three degrees of freedom at each node: translations in the nodal x, y, and z directions." Tet10 has a same problem as Tet4, the solutions in shear components appear not good. The plots also show that the solutions with Stress-Recovery-Algorithms has a better performance than the solutions without Stress-Recovery-Algorithm.

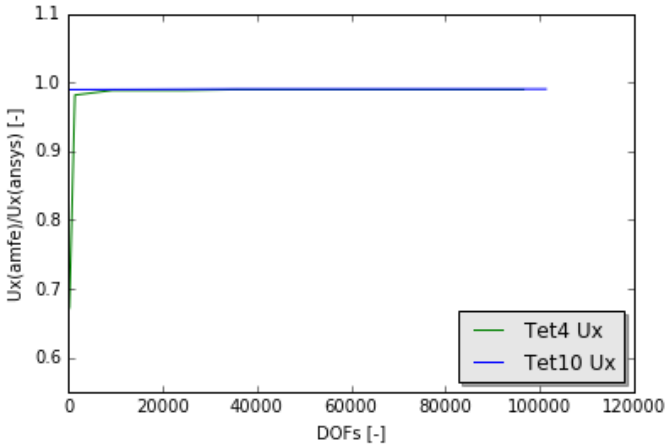

Figure 4.10: Convergence plot of Tet4 and Tet10

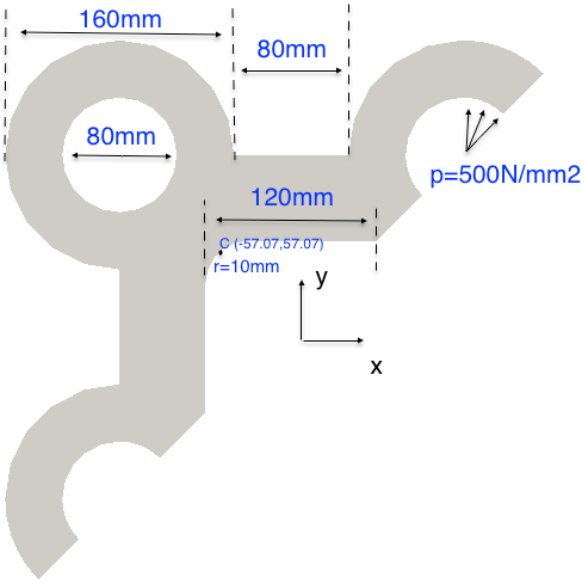

Figure 4.11: Convergence analysis of Von Mises stress in a 2D model

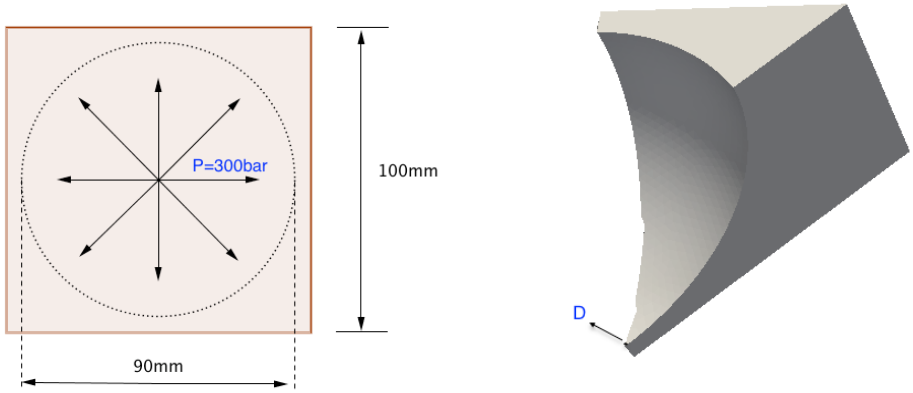

Figure 4.12: Left: Top view of the tank; Right: the 1/16 model of the tank, point D is regarded as the observer point for analysis

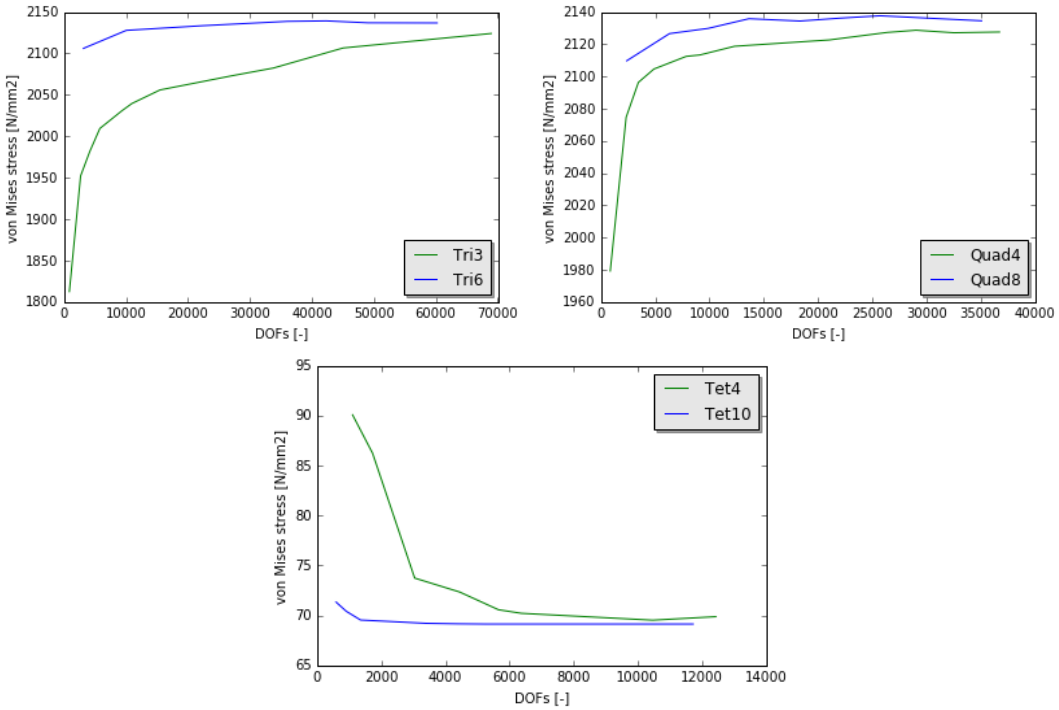

Figure 4.13: Convergence analysis of Von Mises stress, the analytical solutions are $2147.43 N/mm^2$ for 2D model and $68.94 N/mm^2$ for 3D model

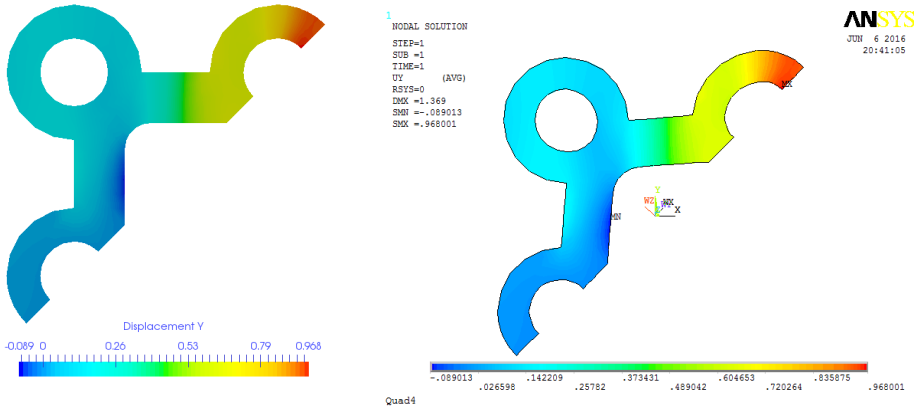

Figure 4.14: Chromatic aberration between ParaView and ANSYS

Table 4.3: Convergence analysis of displacement for triangular element

DOFs	Tri3-Ux	Tri3-Uy	DOFs	Tri6-Ux	Tri6-Uy
90	-0.000873929	0.0068367	62	-0.00132675	0.00847992
292	-0.00127231	0.00993406	114	-0.00149616	0.011506
1038	-0.00143843	0.0112263	138	-0.00149623	0.011513
3898	-0.00148735	0.0112263	292	-0.001504	0.0117277
4170	-0.00148835	0.0116131	1780	-0.001504	0.0117383
4698	-0.00148919	0.0116189	4830	-0.00150448	0.0117391
5298	-0.00149204	0.0116457	8986	-0.00150449	0.0117392
5842	-0.00149269	0.0116505	12073	-0.00150450	0.0117392
14322	-0.00149982	0.0117028	15609	-0.00150451	0.0117393
25634	-0.00150168	0.0117171	19300	-0.00150453	0.0117393

Table 4.4: Convergence analysis of displacement for quadrilateral element

DOFs	Quad4-Ux	Quad4-Uy	DOFs	Quad8-Ux	Quad8-Uy
48	-0.00078276	0.00620667	142	-0.00120593	0.00847992
372	-0.0014393	0.0112254	286	-0.00139123	0.0108697
1332	-0.00147509	0.0115086	466	-0.00139228	0.0108725
3374	-0.00149431	0.0116894	1090	-0.00147418	0.0115056
4842	-0.00149817	0.0116894	2870	-0.00149483	0.0116652
6228	-0.00149945	0.0116992	6450	-0.00149948	0.0117018
8386	-0.00150076	0.0117105	10346	-0.00150103	0.0117032
10232	-0.00150123	0.0117123	12342	-0.00150163	0.0117056
15344	-0.00150134	0.0117144	15656	-0.00150166	0.0117079
20044	-0.00150143	0.0117151	22344	-0.00150183	0.0117186

Table 4.5: Convergence analysis of displacement for tetrahedral element

DOFs	Tet4-Ux	Tet4-Uy	Tet4-Uz	DOFs	Tet10-Ux	Tet10-Uy	Tet10-Uz
54	6.320	0	0	243	9.370	0	0
243	6.372	0	0	579	9.370	0	0
1395	9.295	0	0	13857	9.372	0	0
9315	9.360	0	0	36312	9.378	0	0
24579	9.360	0	0	43521	9.378	0	0
34632	9.370	0	0	57537	9.378	0	0
53307	9.370	0	0	60299	9.378	0	0
64343	9.370	0	0	79683	9.378	0	0
76443	9.370	0	0	84322	9.378	0	0
96567	9.370	0	0	101321	9.378	0	0

Table 4.6: Convergence analysis of Von Mises stress

DOFs	Tri3	DOFs	Tri6	DOFs	Quad4	DOFs	Quad8	DOFs	Tet4	DOFs	Tet
870	1812.47	3132	2106.11	854	1979.22	2372	2109.74	1101	90.07	588	71.3
2682	1952.20	10058	2127.91	2320	2074.70	6314	2126.62	1725	86.24	906	70.4
4172	1981.88	15892	2130.58	3448	2096.34	9890	2129.87	3039	73.73	1352	69.5
5788	2009.47	22286	2133.51	4882	2104.64	13648	2135.88	4440	72.34	2258	69.3
9354	2030.88	36028	2138.76	7856	2112.48	18340	2134.44	5652	70.57	3396	69.2
10952	2039.58	42344	2139.34	9128	2113.38	22036	2136.23	6366	70.21	4351	69.1
15490	2055.98	49044	2137.02	12310	2118.76	25724	2137.70	10461	69.52	5243	69.1
27332	2073.44	60214	2136.87	21060	2122.70	35096	2134.58	12342	69.87	11721	69.1
33770	2082.39	-	-	26384	2127.35	-	-	-	-	-	-
45032	2106.63	-	-	29056	2128.73	-	-	-	-	-	-
68943	2124.15	-	-	32548	2128.16	-	-	-	-	-	-
-	-	-	-	36750	2128.62	-	-	-	-	-	-

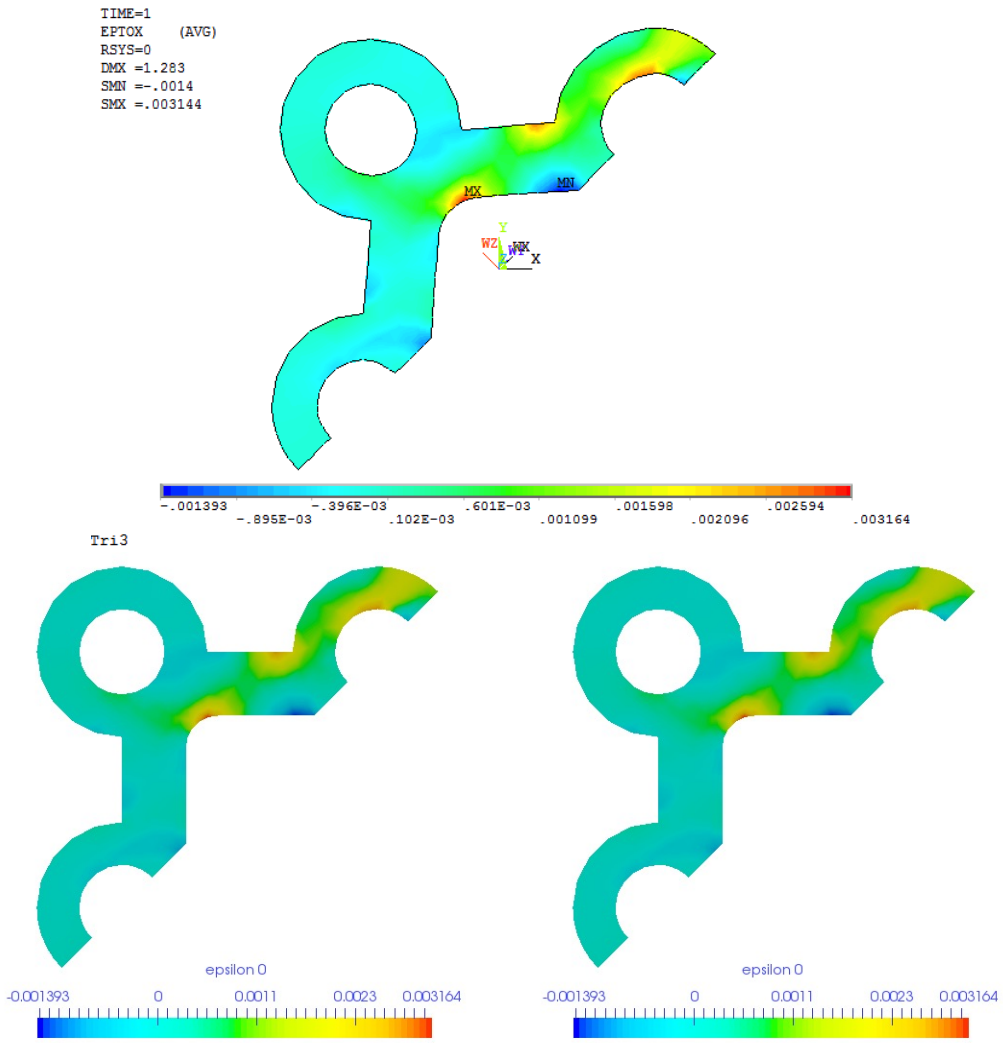

Figure 4.15: Mesh with Tri3, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

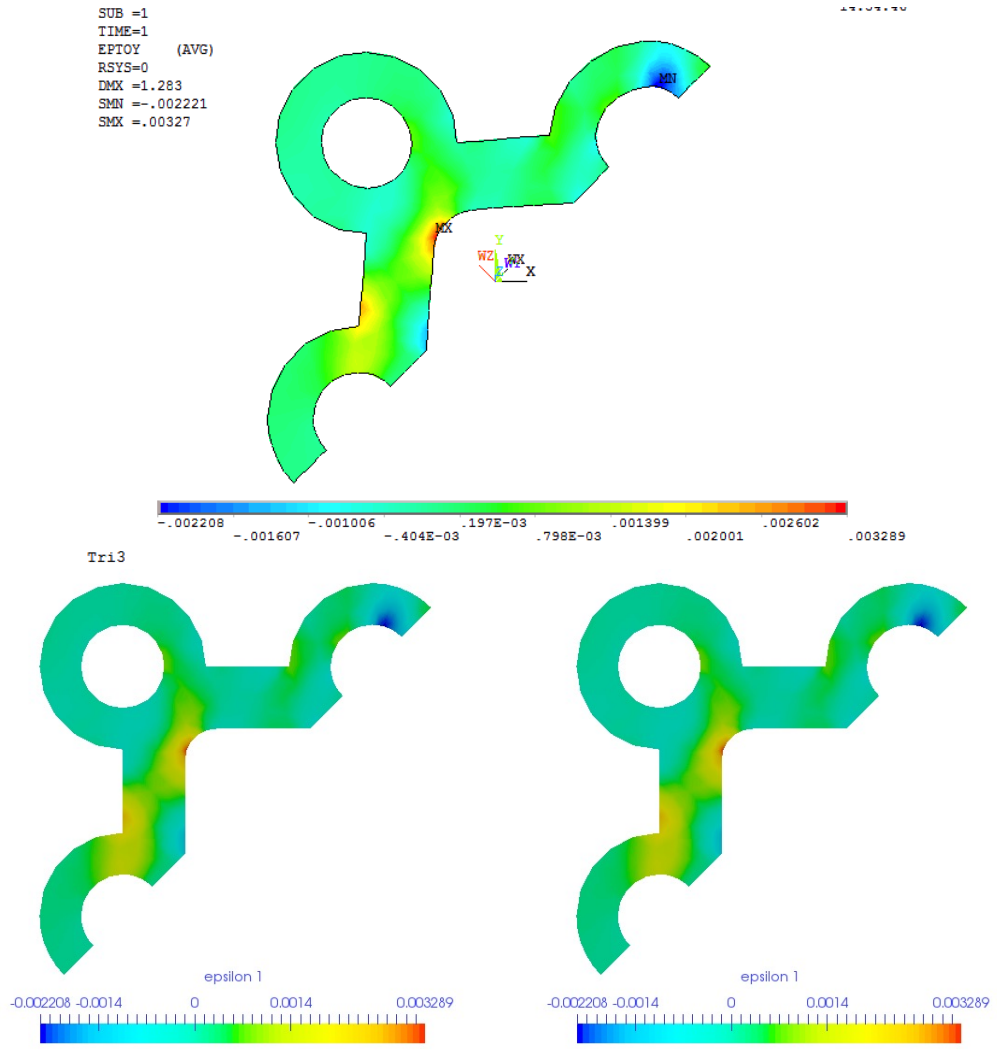

Figure 4.16: Mesh with Tri3, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

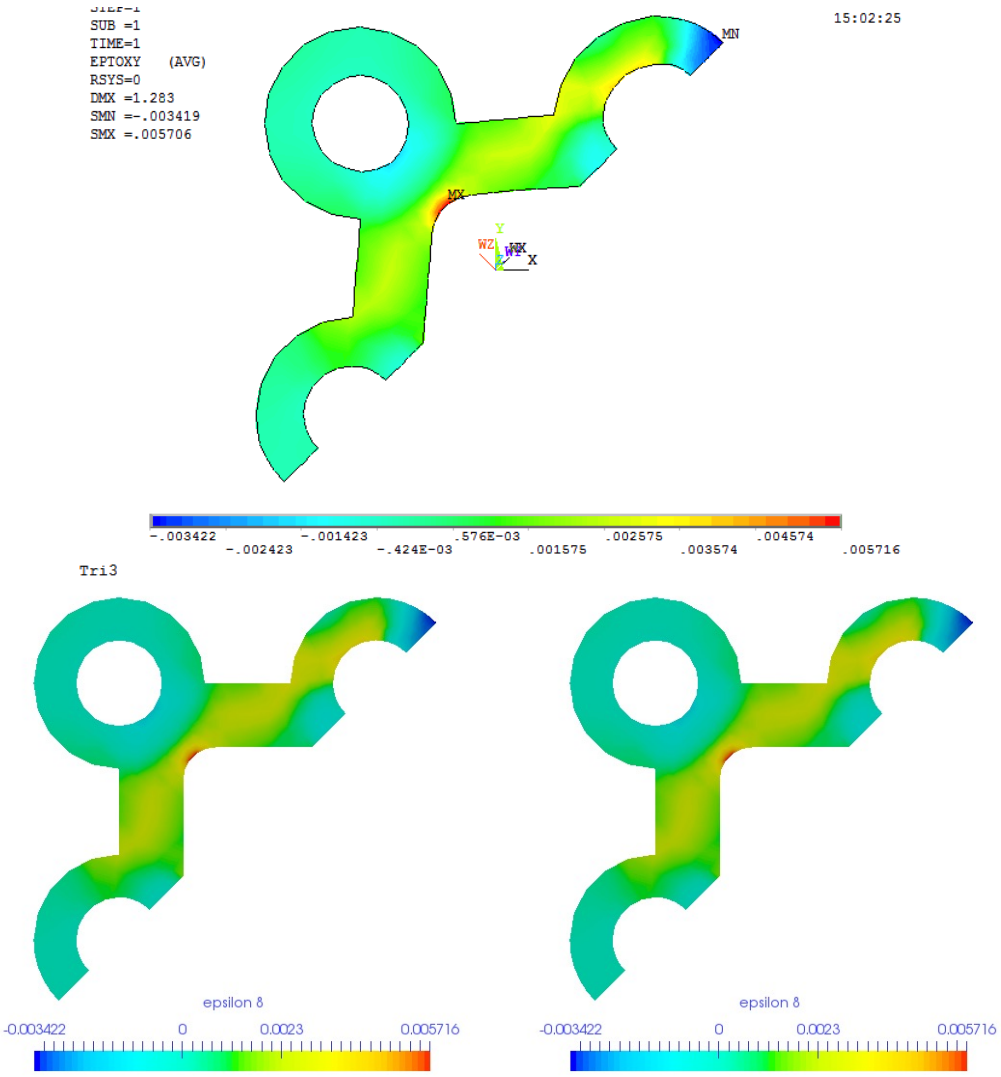

Figure 4.17: Mesh with Tri3, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

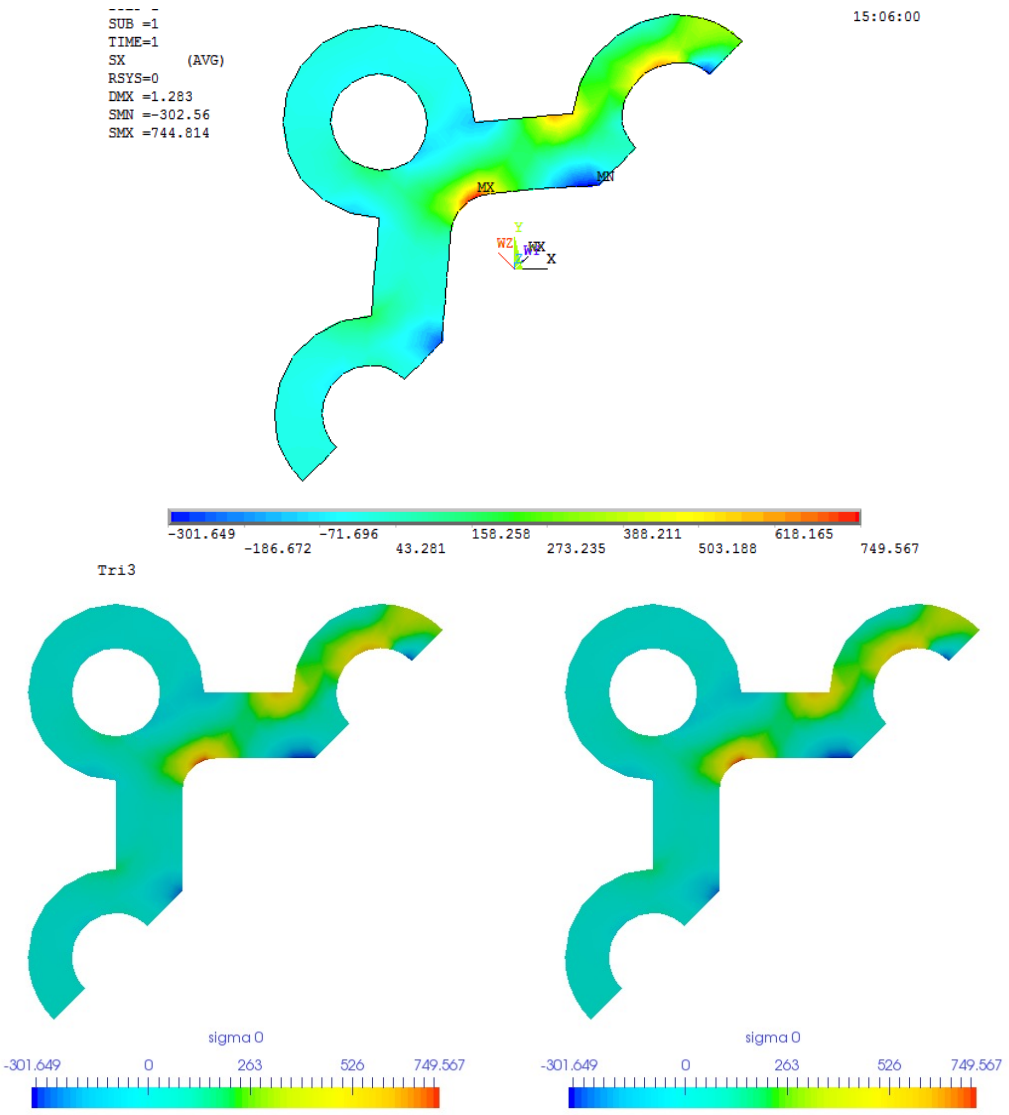

Figure 4.18: Mesh with Tri3, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

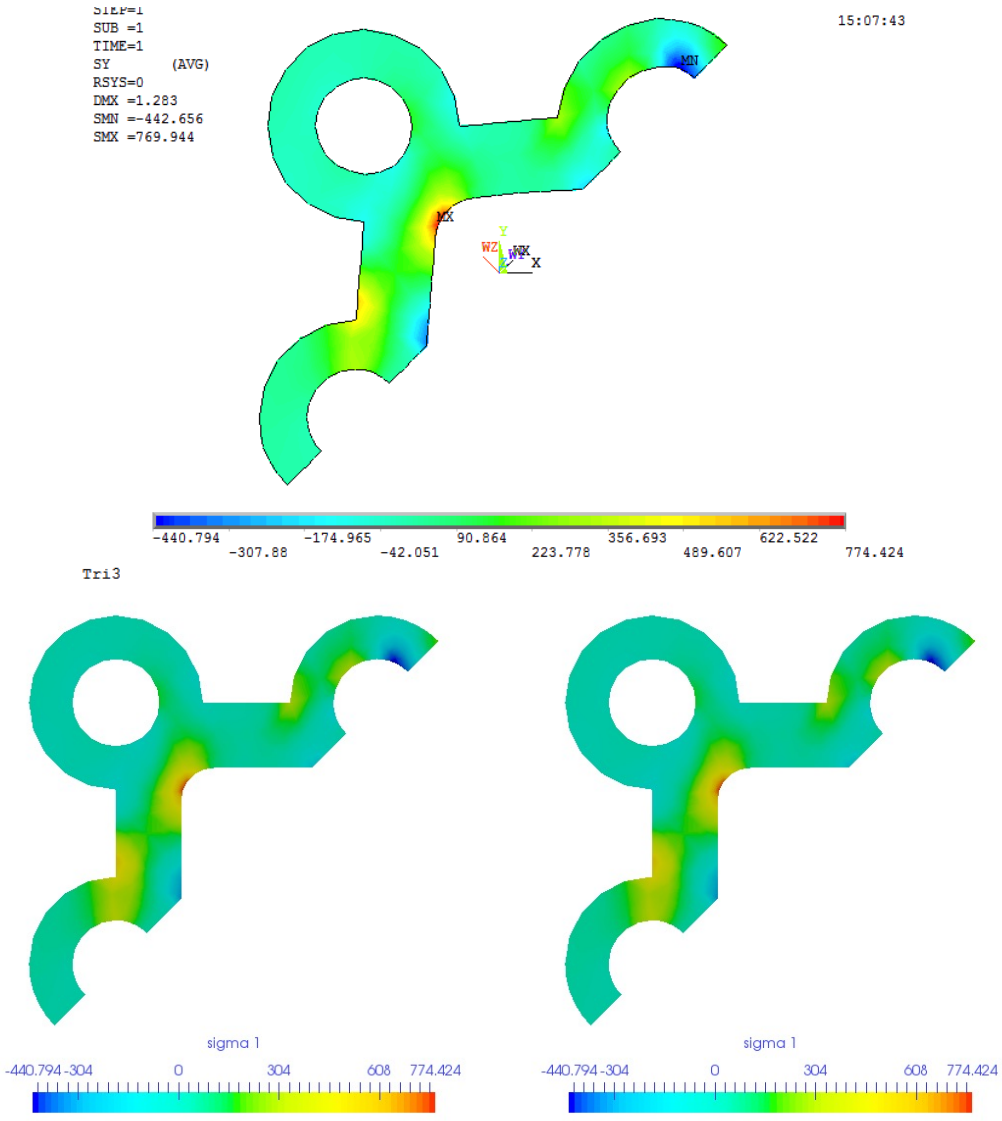

Figure 4.19: Mesh with Tri3, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

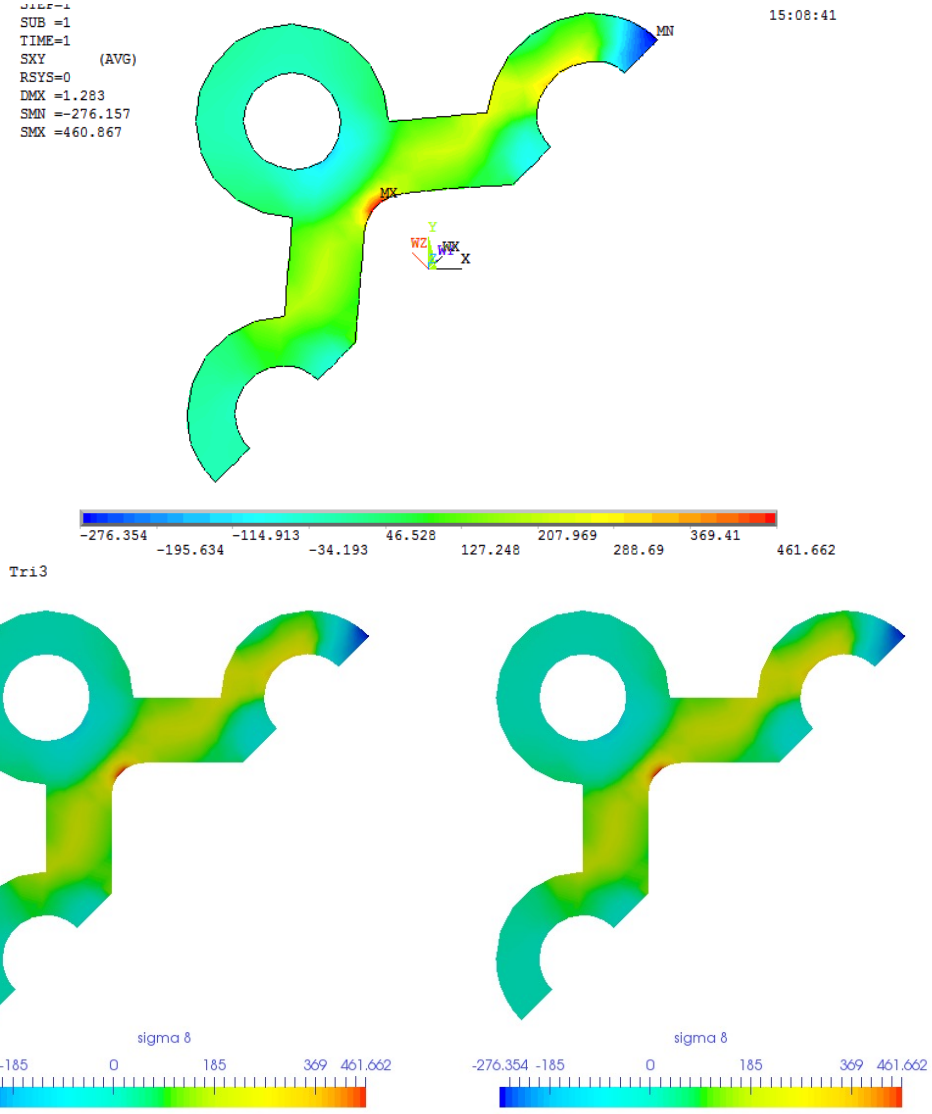

Figure 4.20: Mesh with Tri3, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

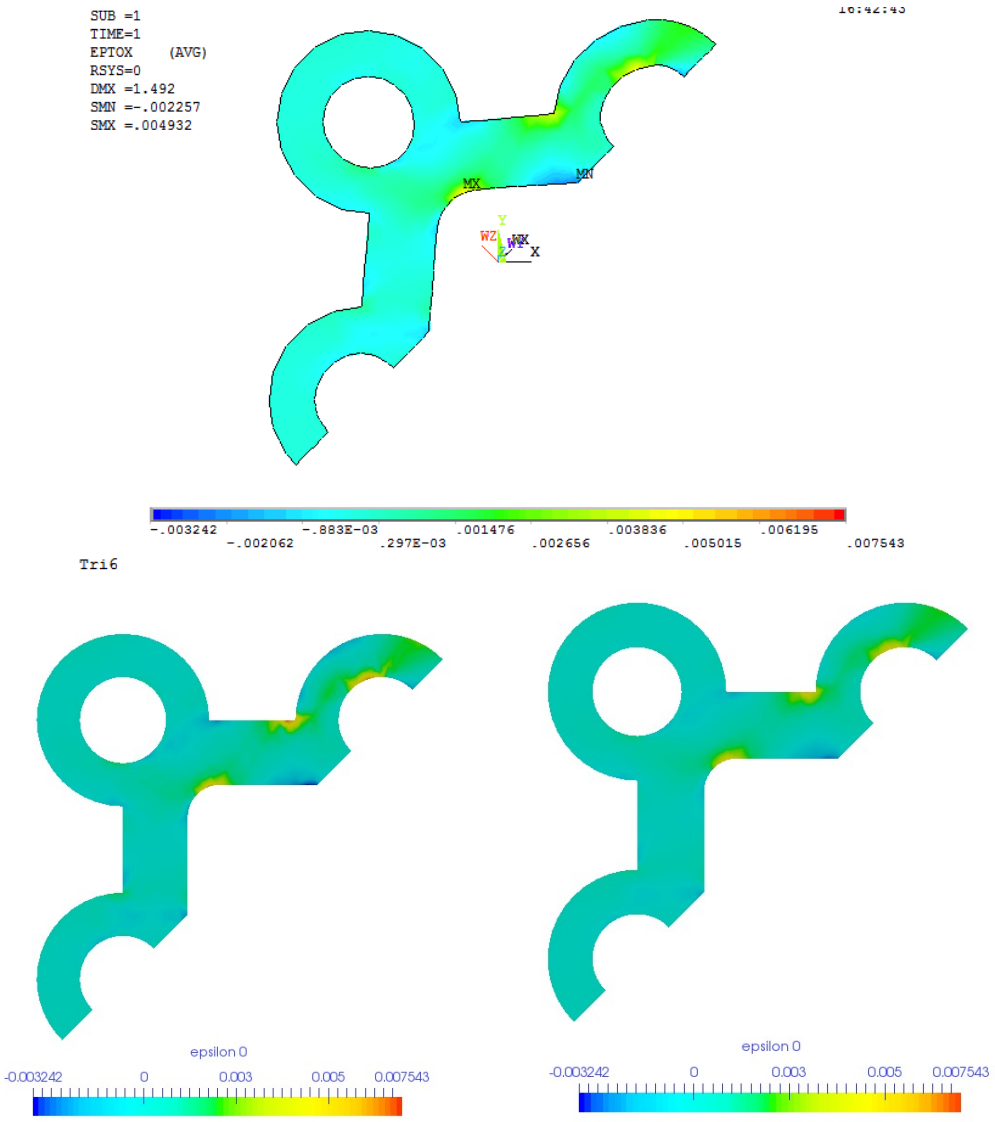

Figure 4.21: Mesh with Tri6, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

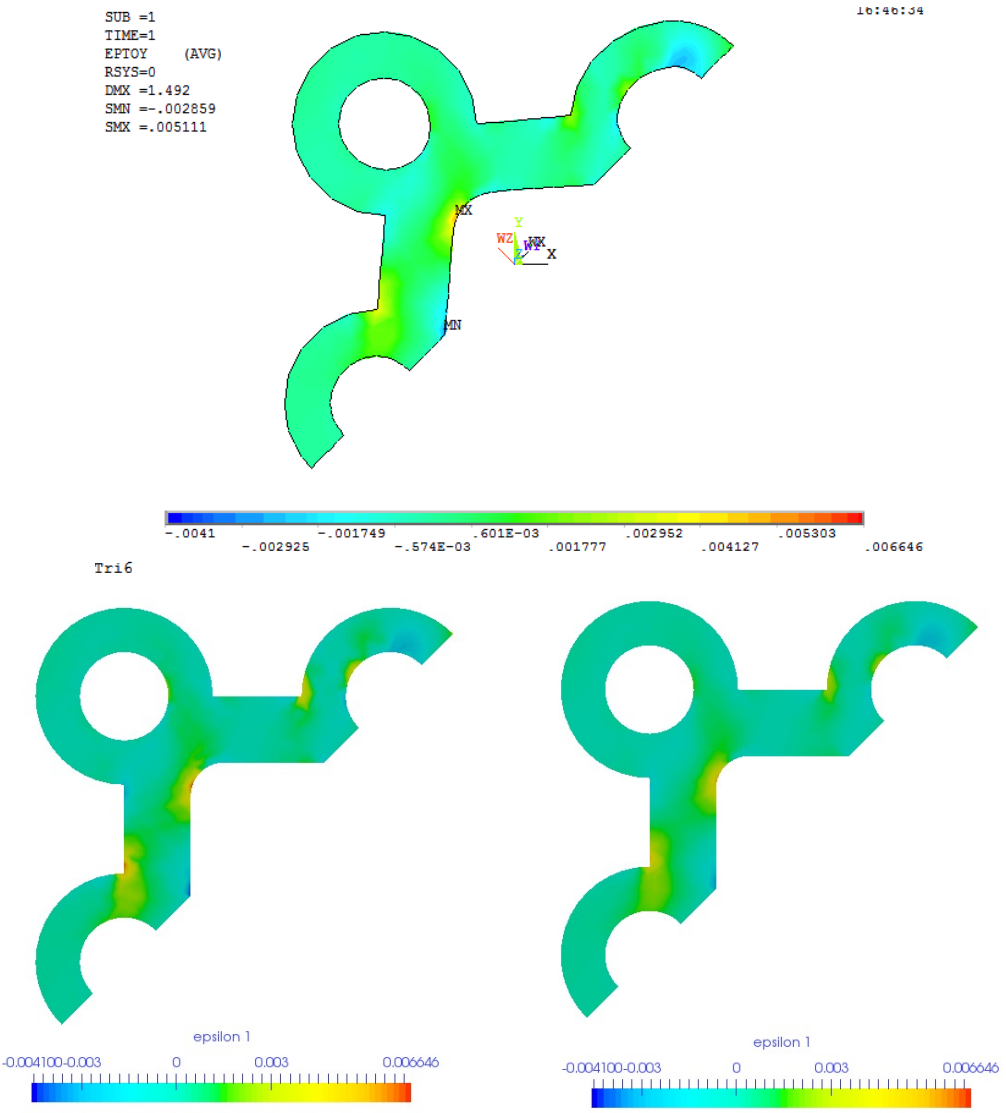

Figure 4.22: Mesh with Tri6, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

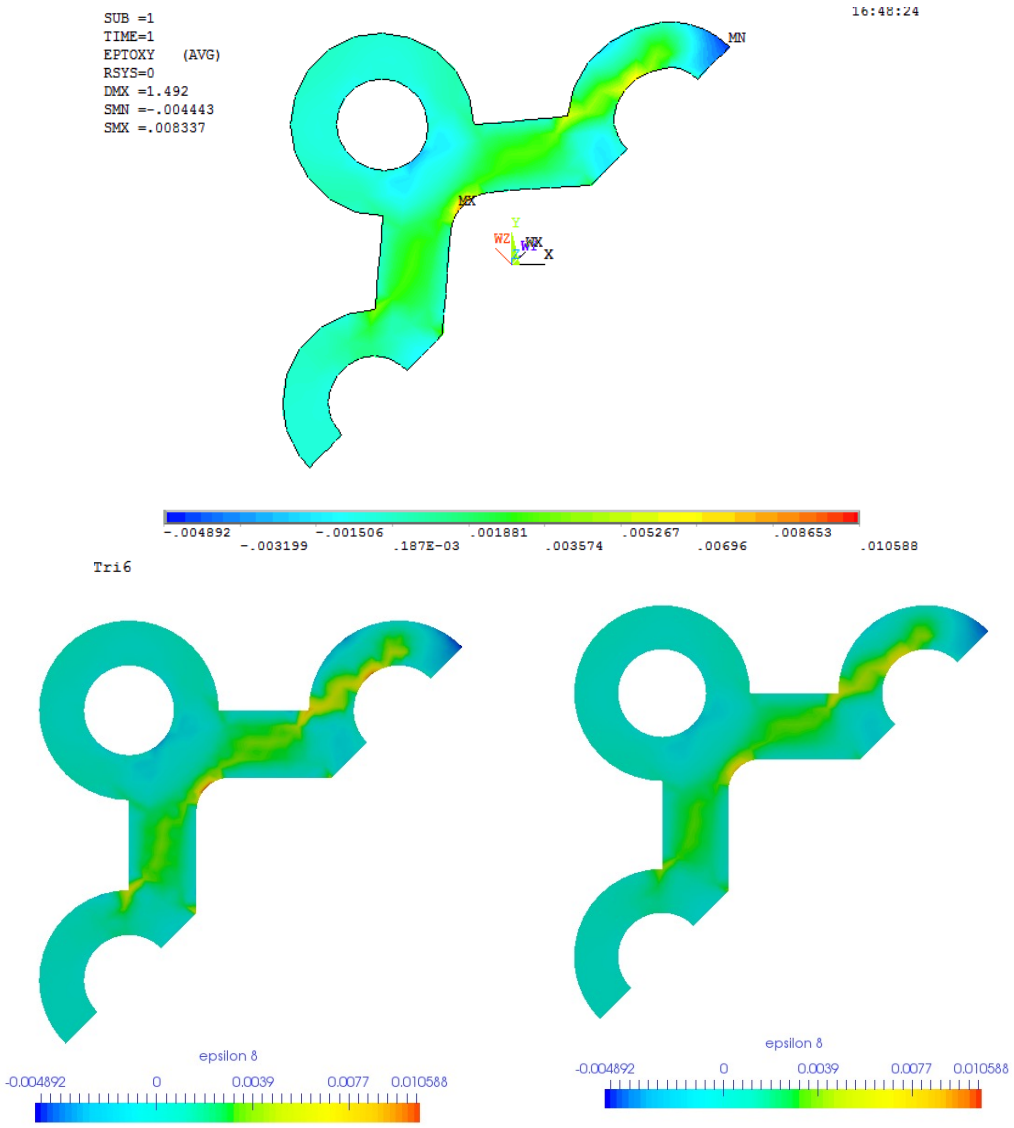

Figure 4.23: Mesh with Tri6, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

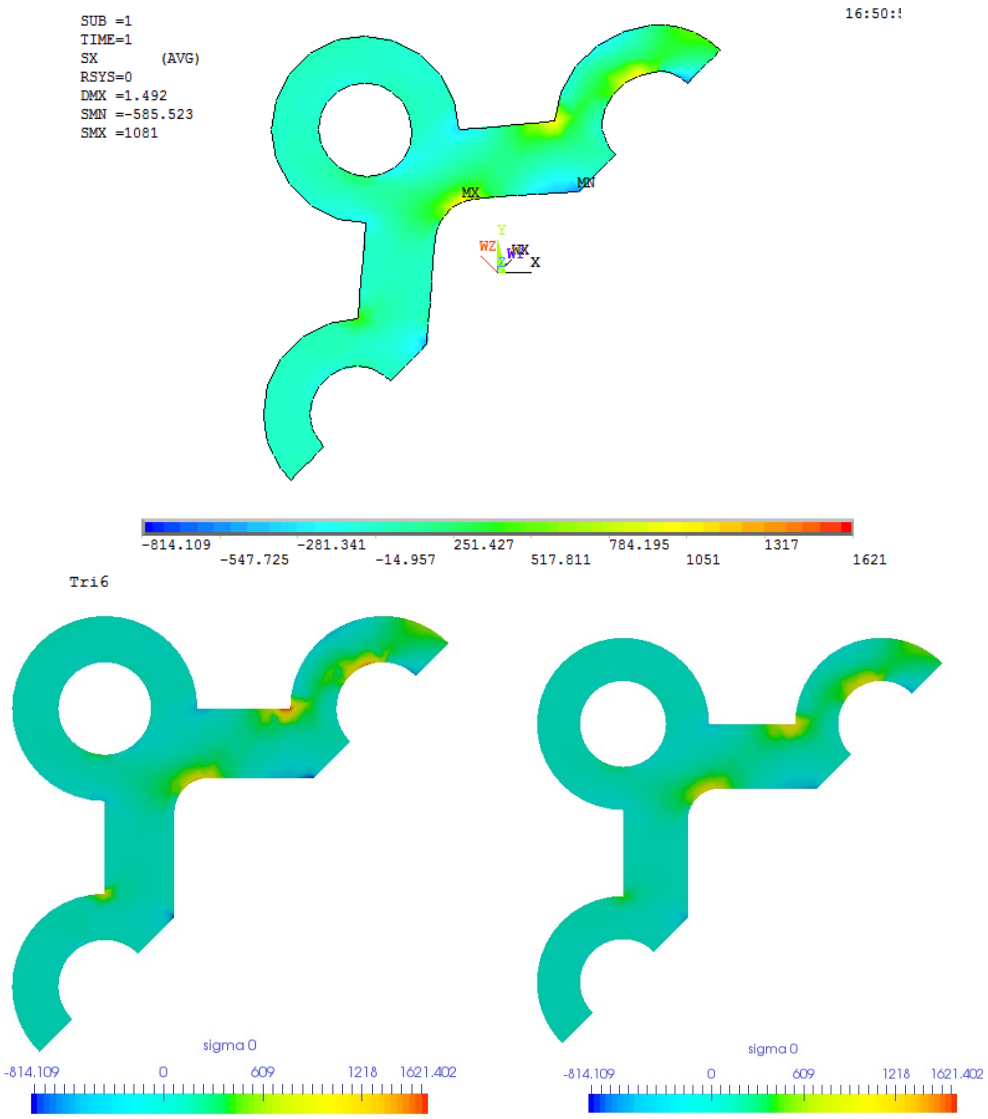

Figure 4.24: Mesh with Tri6, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

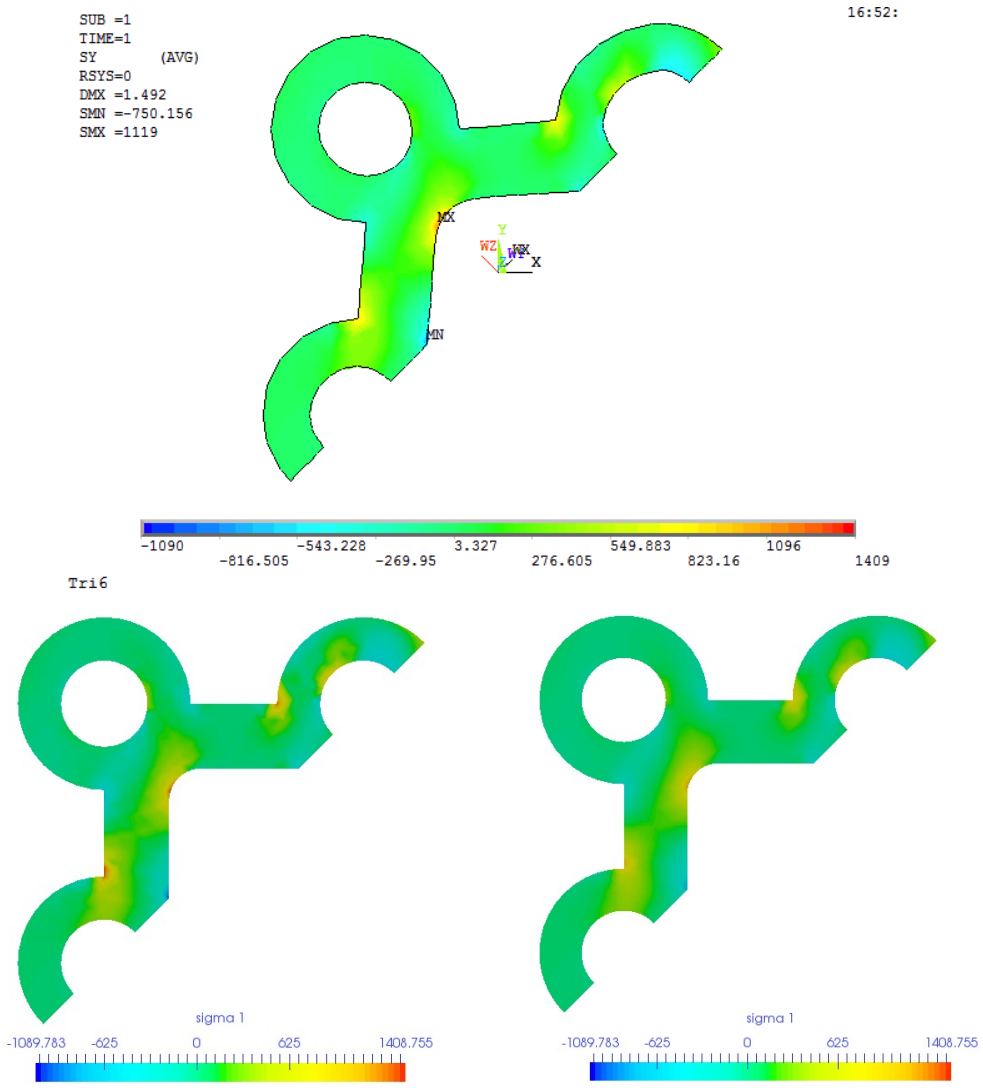

Figure 4.25: Mesh with Tri6, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

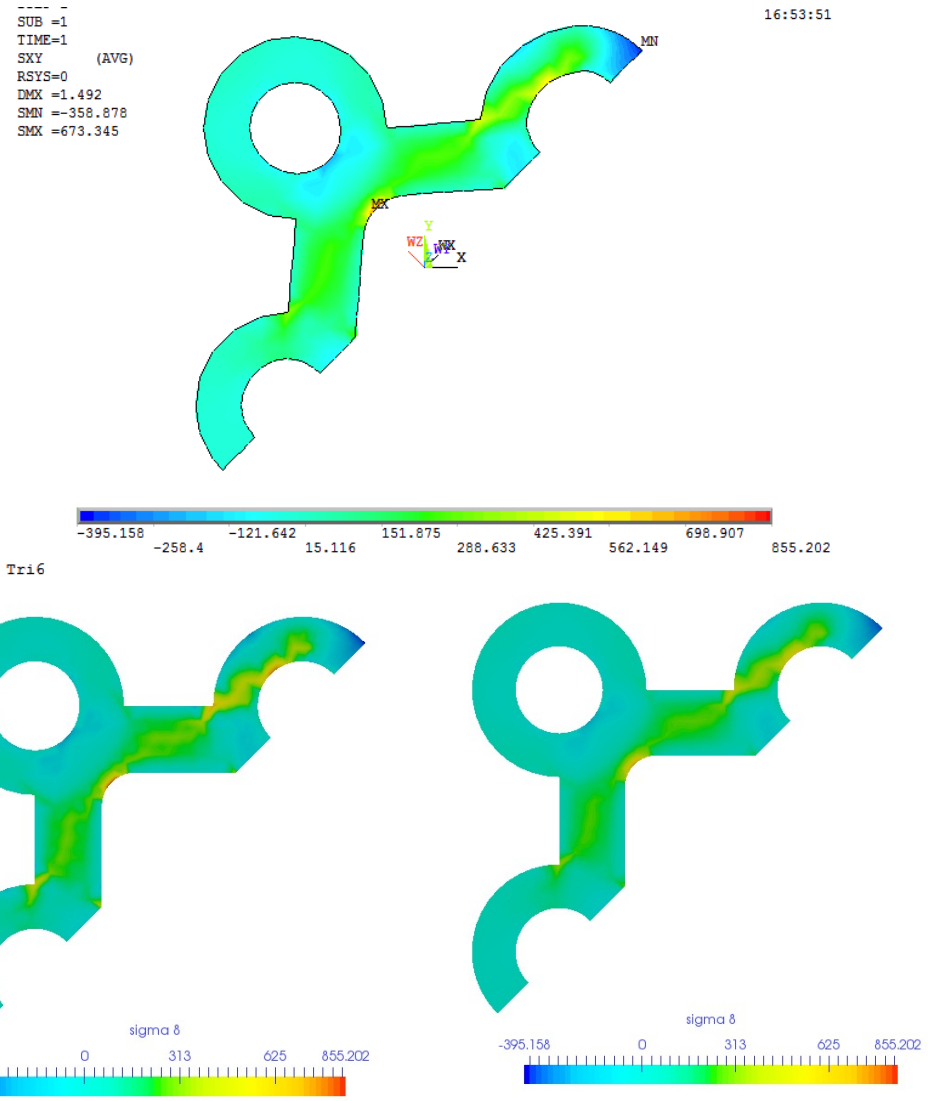

Figure 4.26: Mesh with Tri6, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

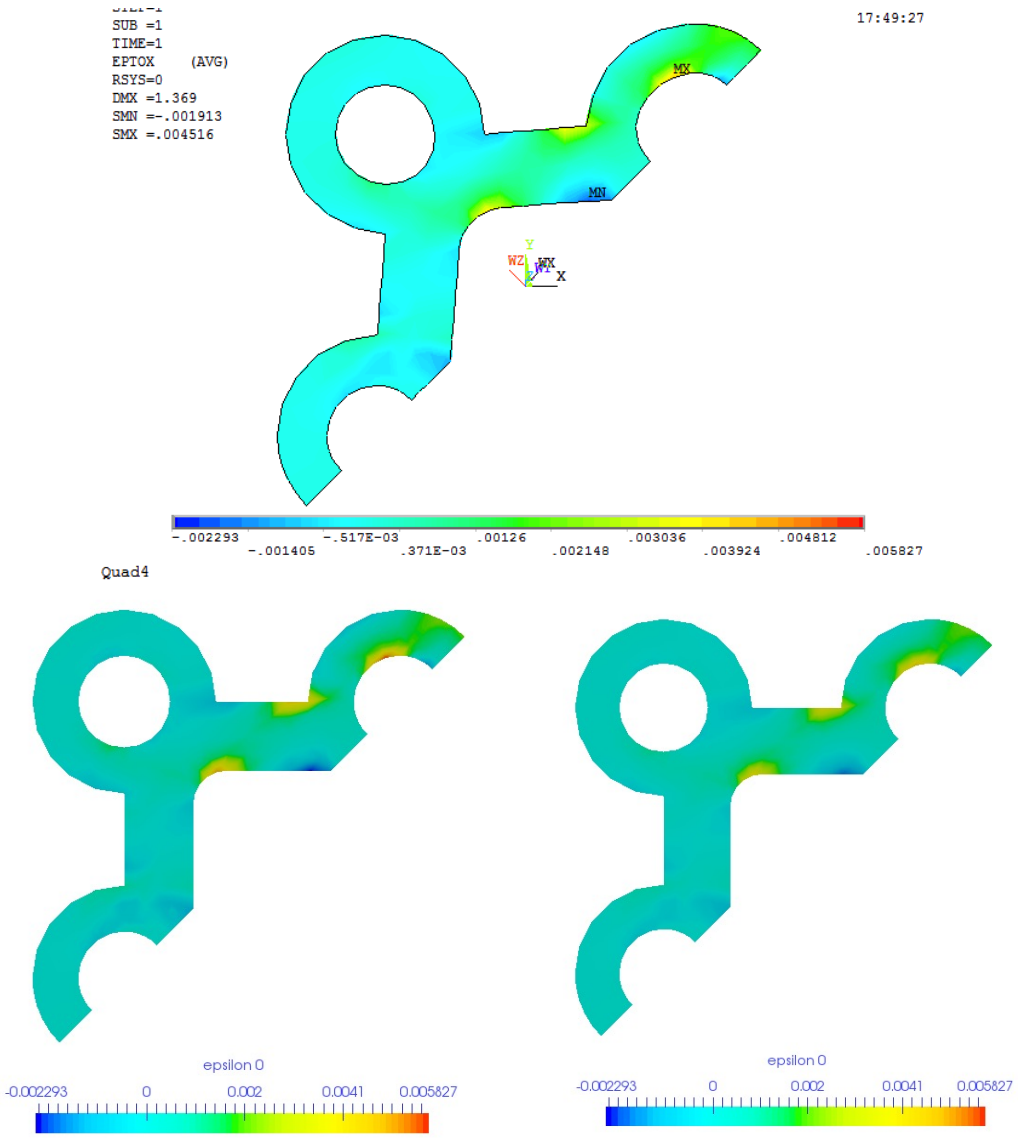

Figure 4.27: Mesh with Quad4, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

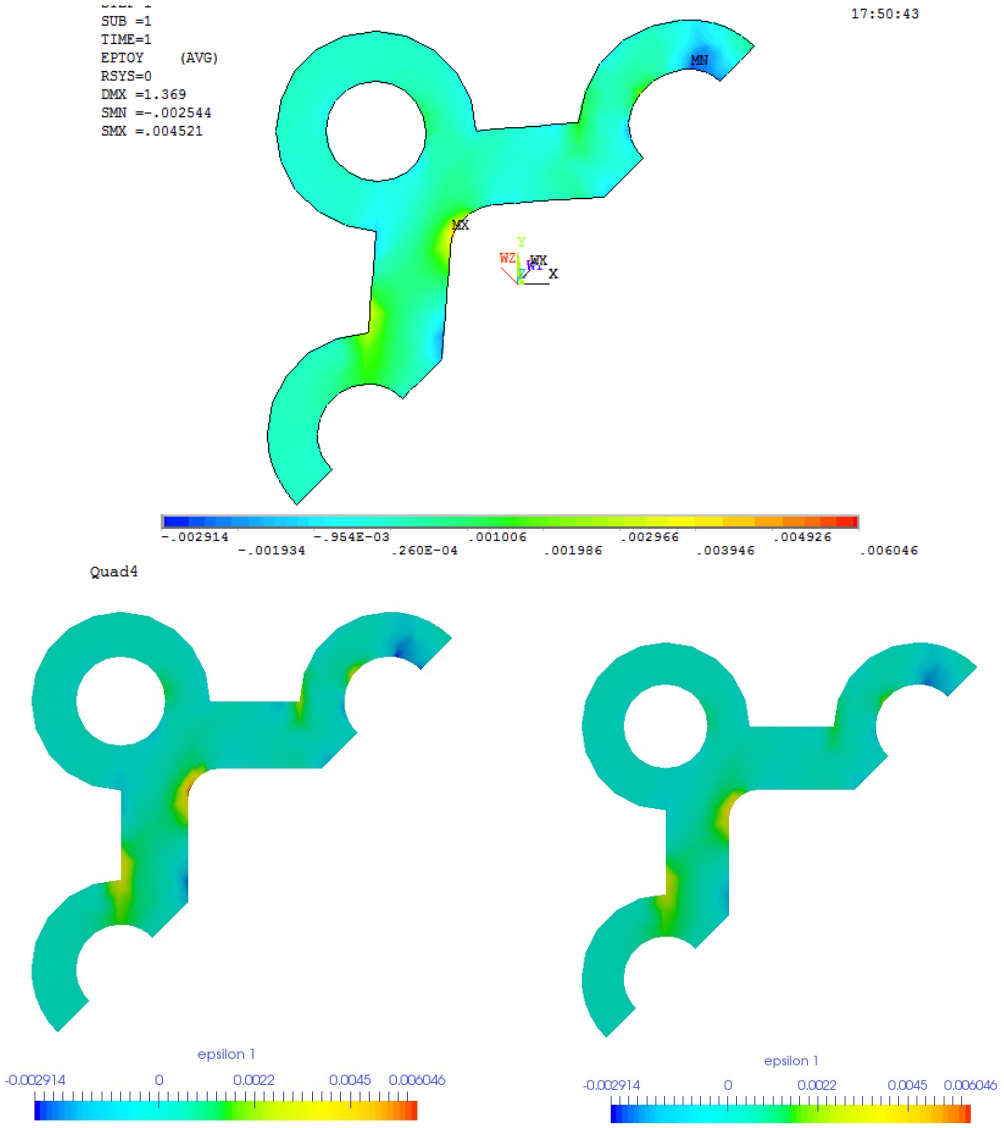

Figure 4.28: Mesh with Quad4, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

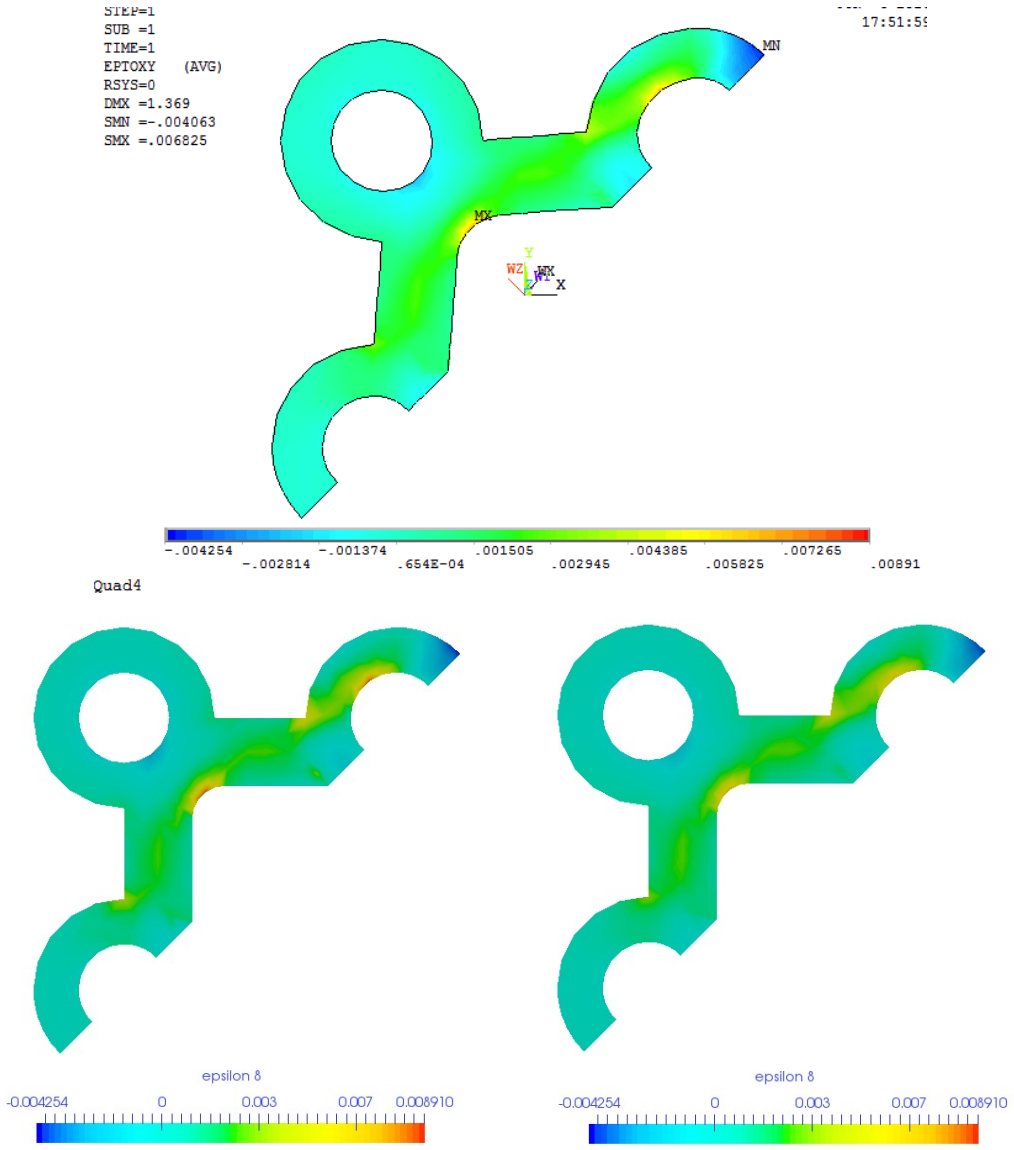

Figure 4.29: Mesh with Quad4, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

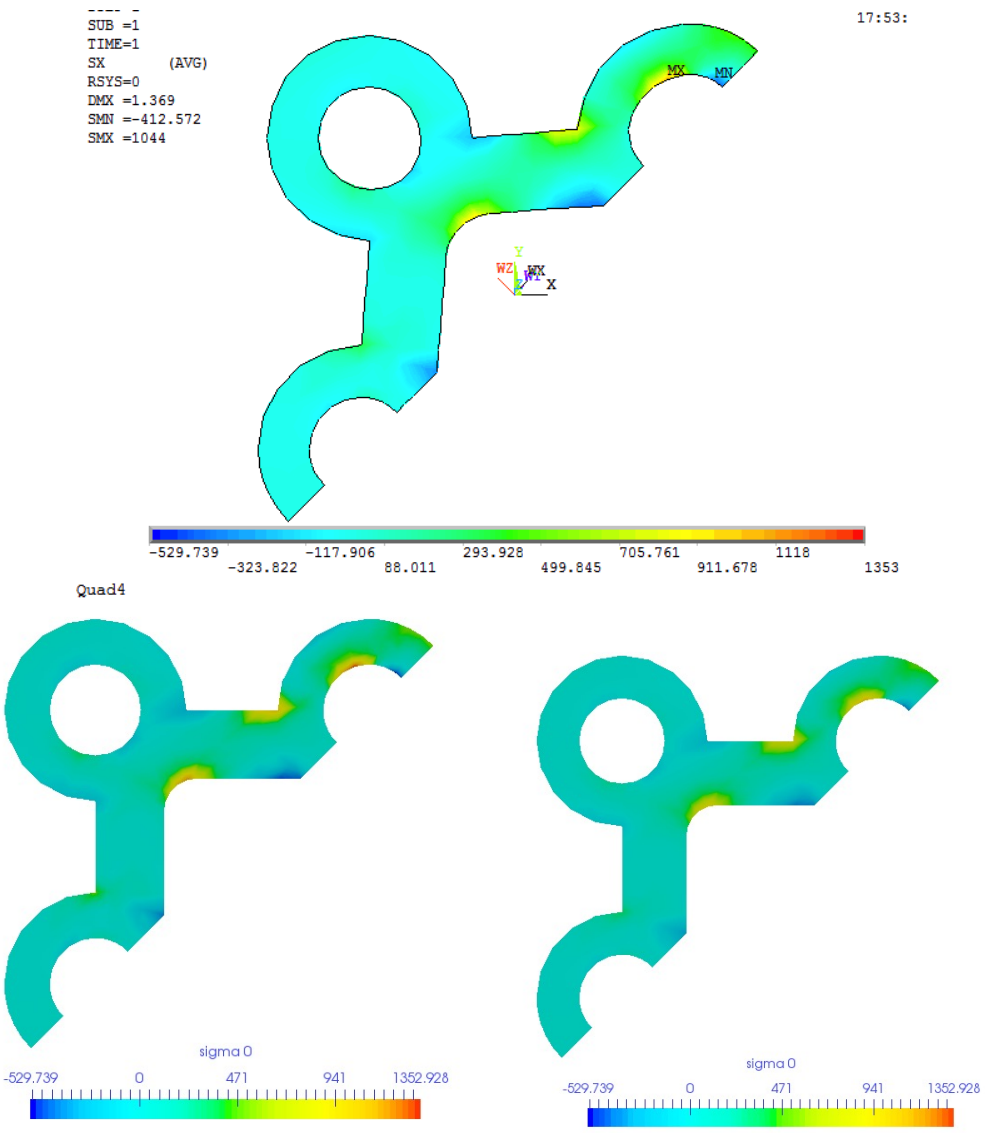

Figure 4.30: Mesh with Quad4, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

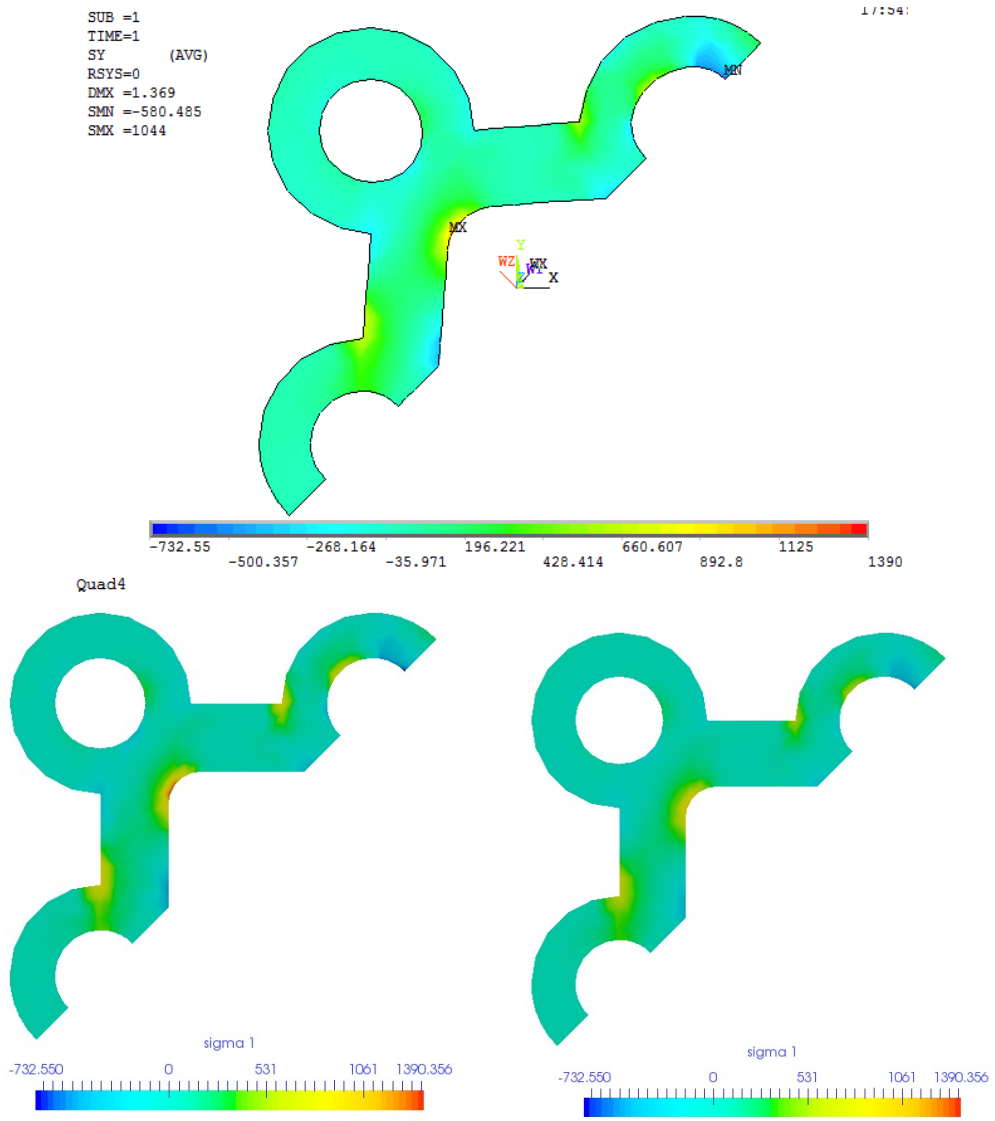

Figure 4.31: Mesh with Quad4, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

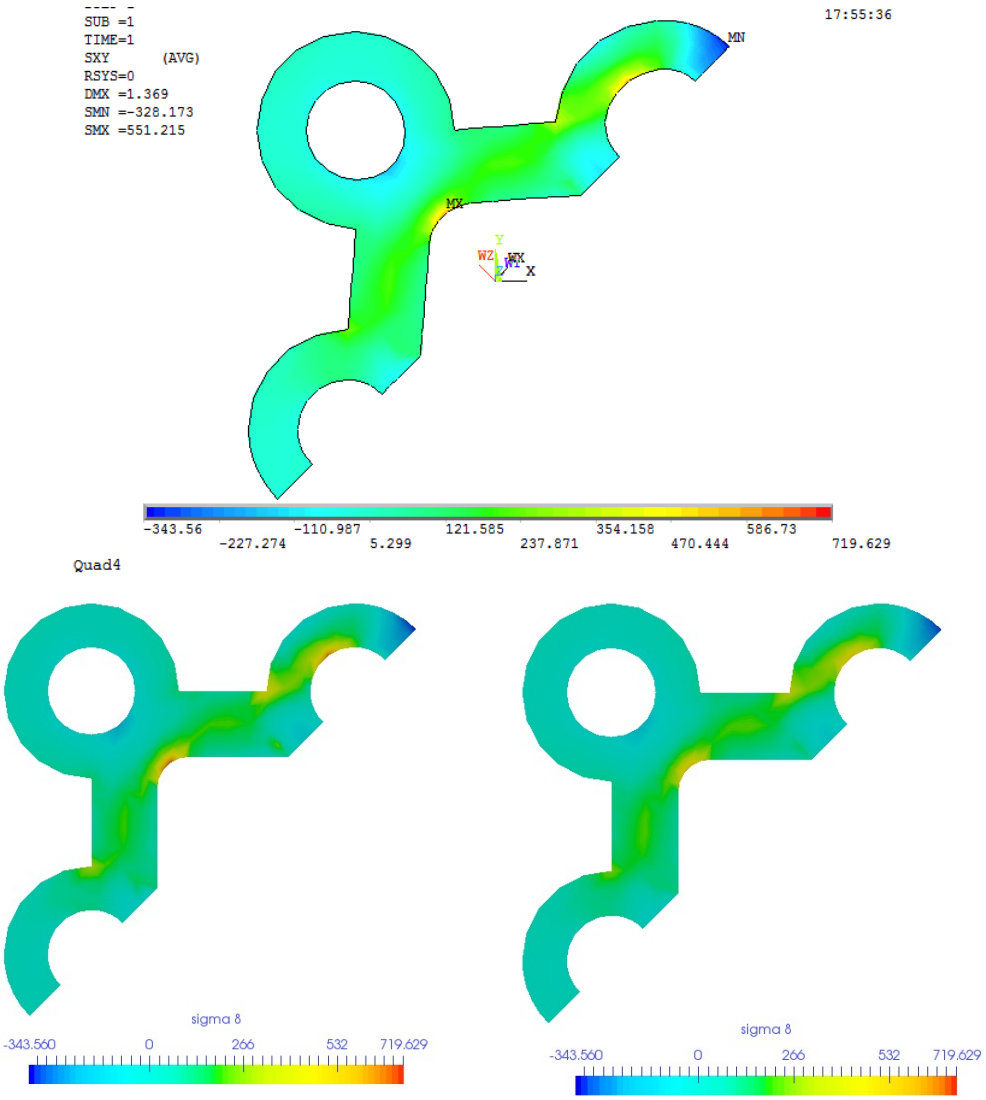

Figure 4.32: Mesh with Quad4, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

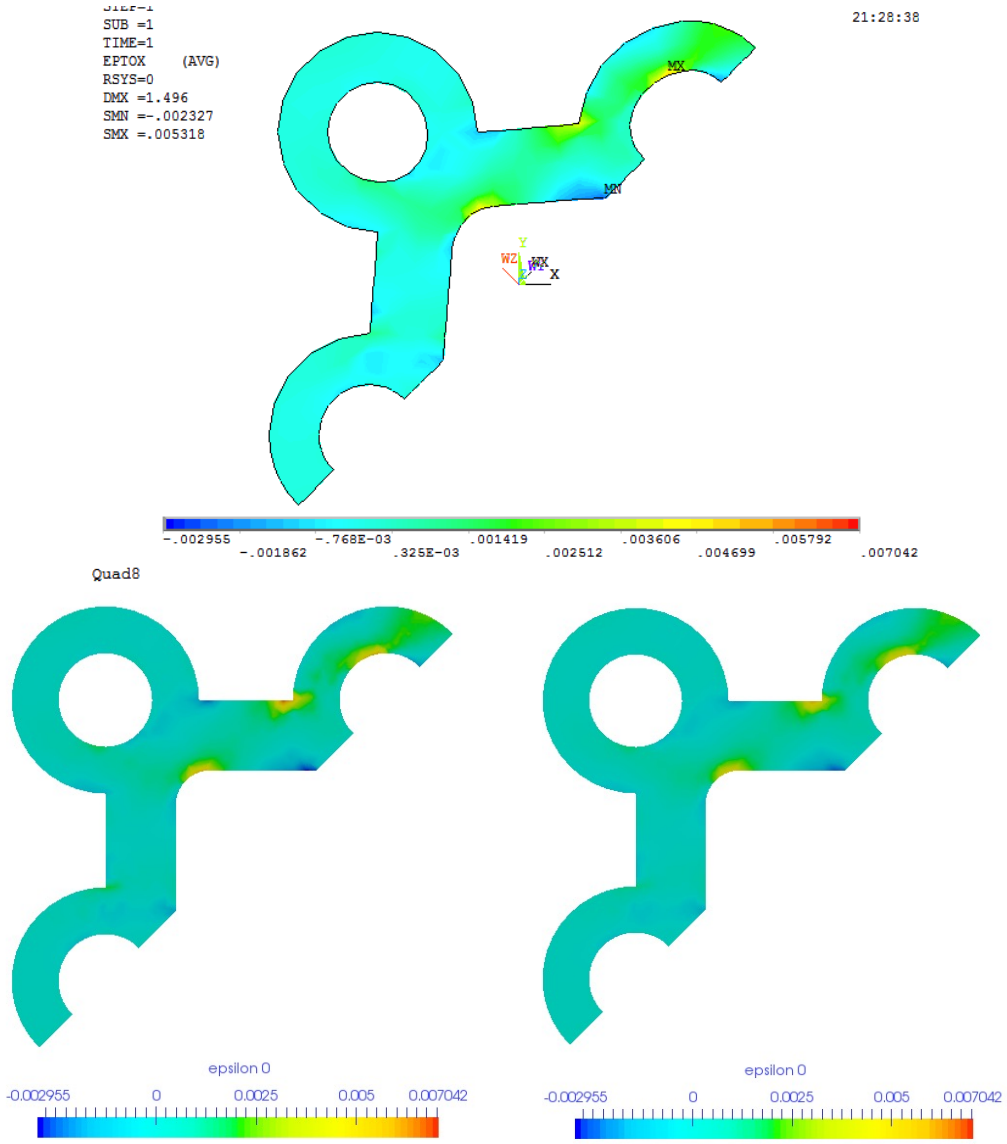

Figure 4.33: Mesh with Quad8, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

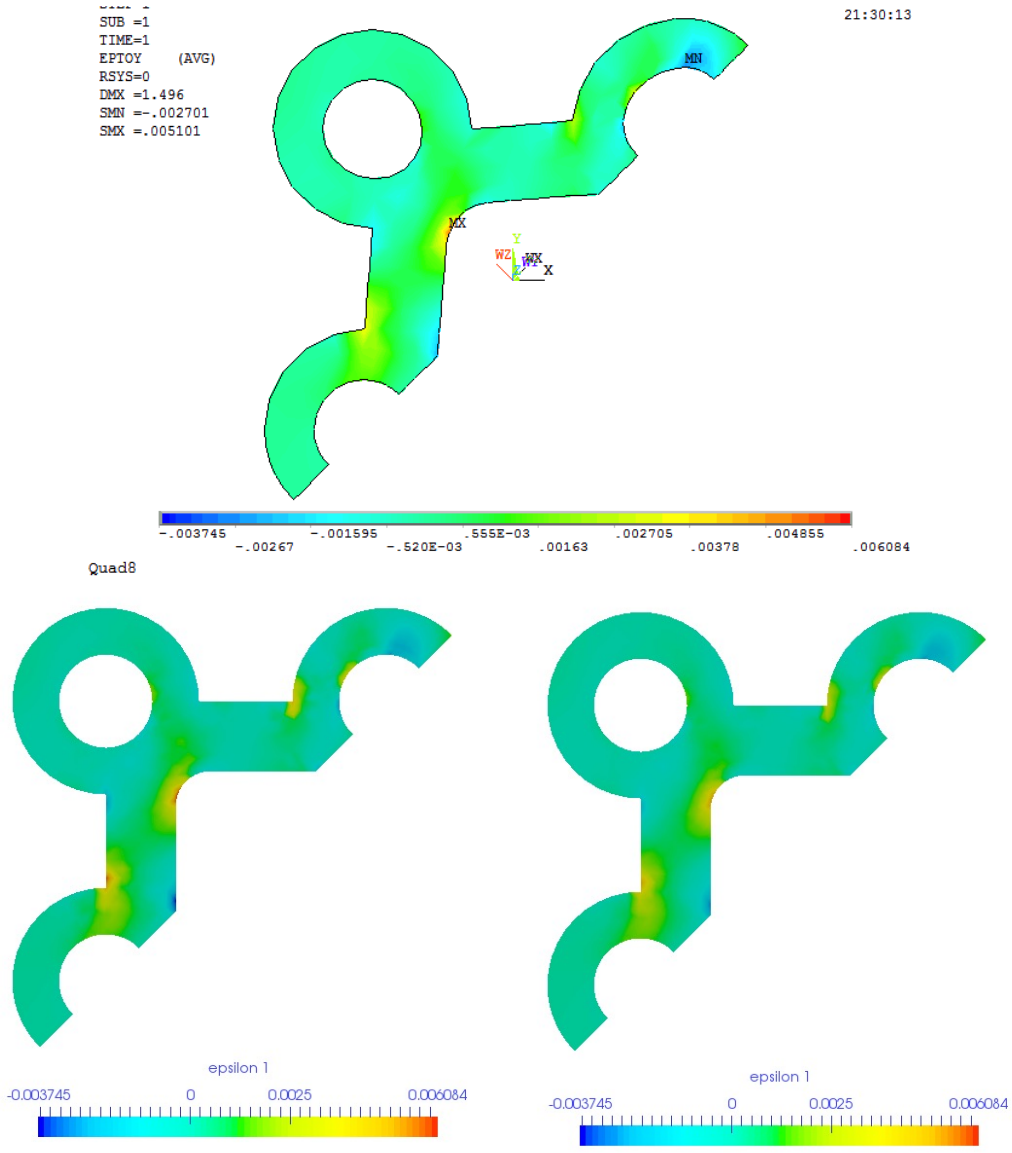

Figure 4.34: Mesh with Quad8, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

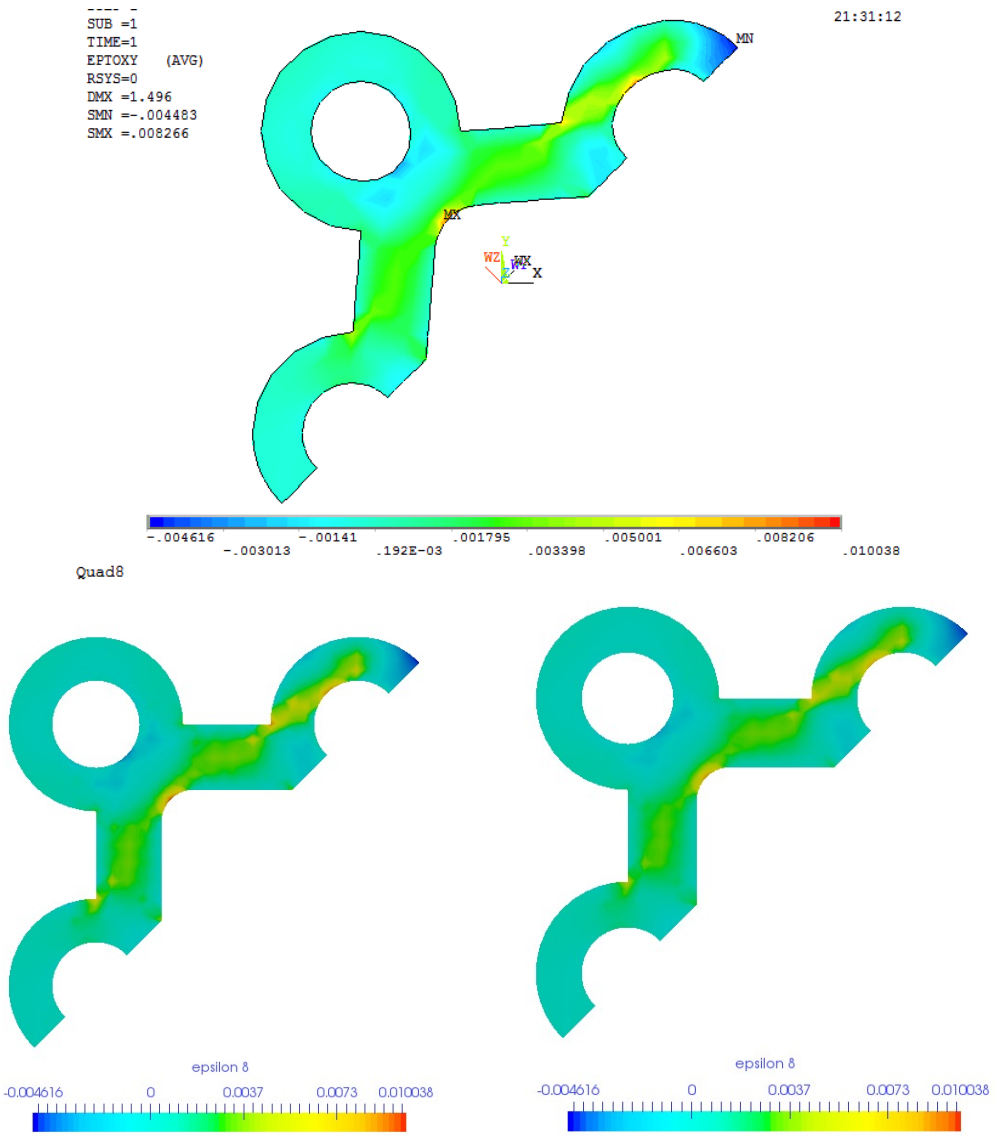

Figure 4.35: Mesh with Quad8, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

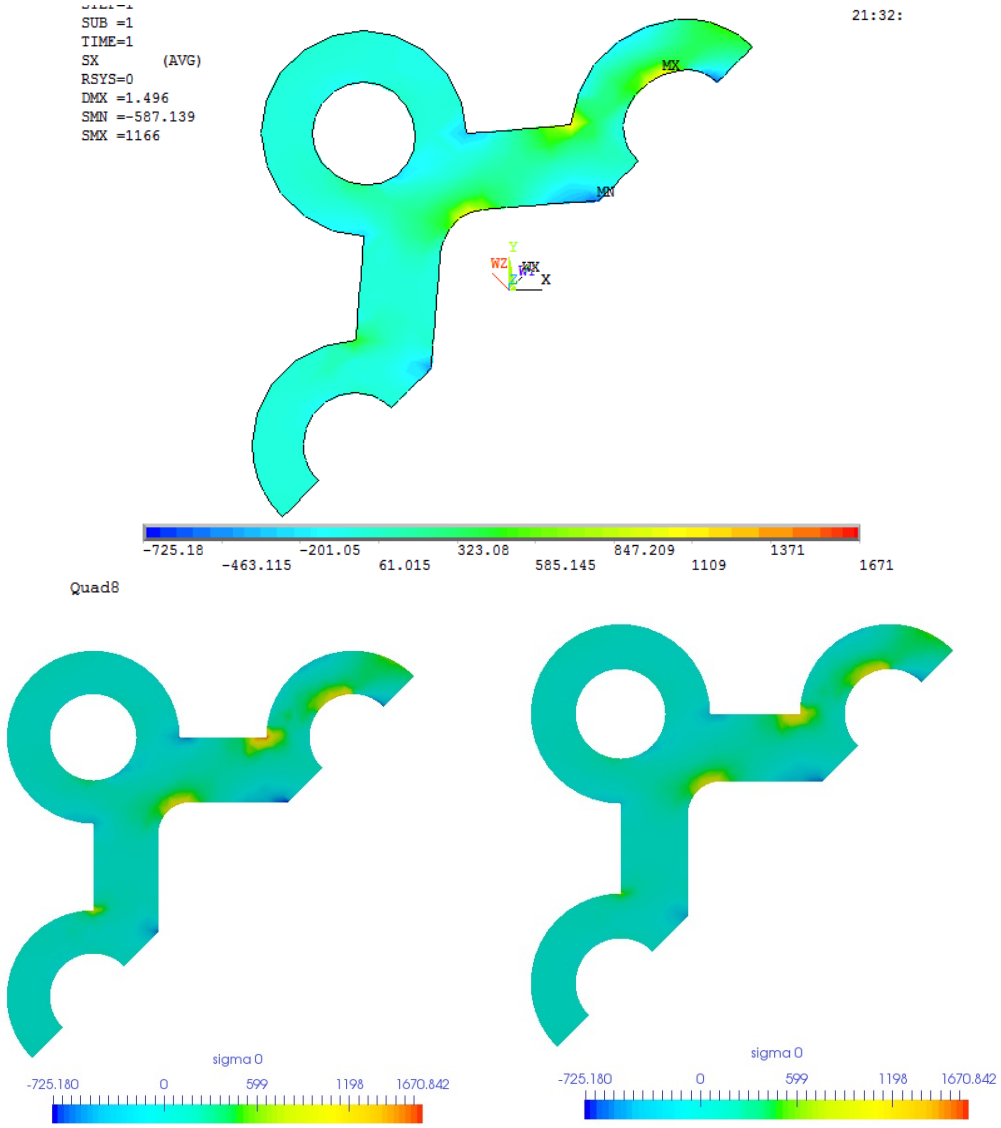

Figure 4.36: Mesh with Quad8, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

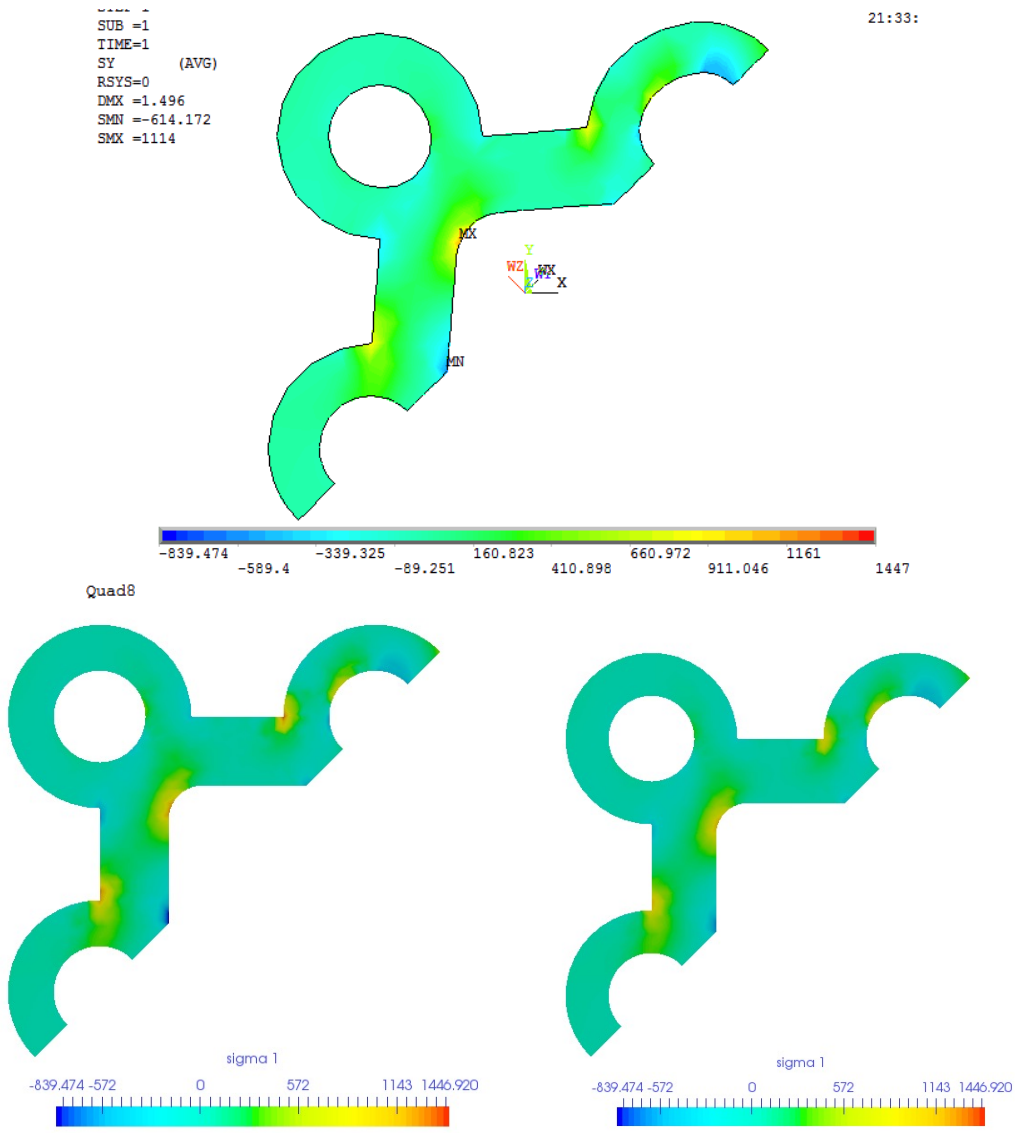

Figure 4.37: Mesh with Quad8, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

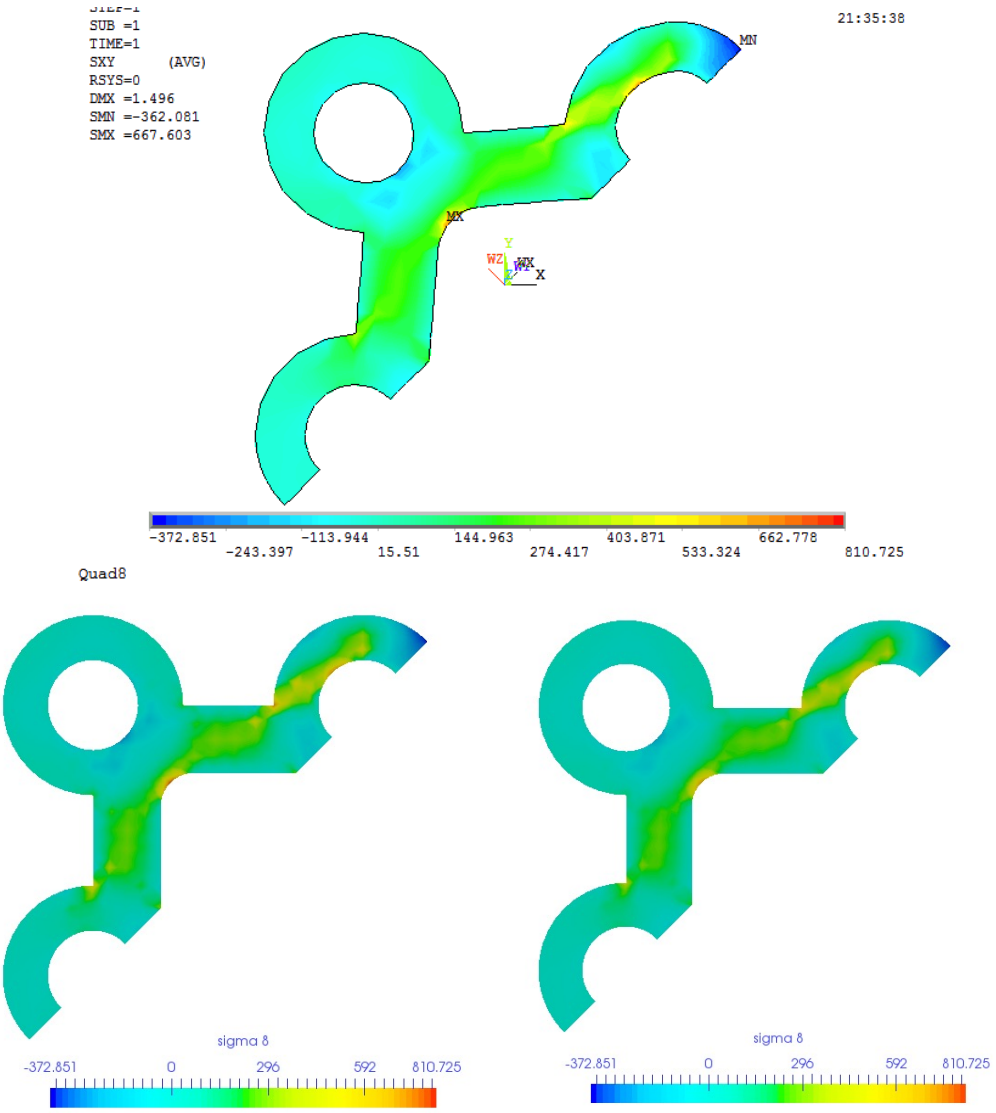

Figure 4.38: Mesh with Quad8, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

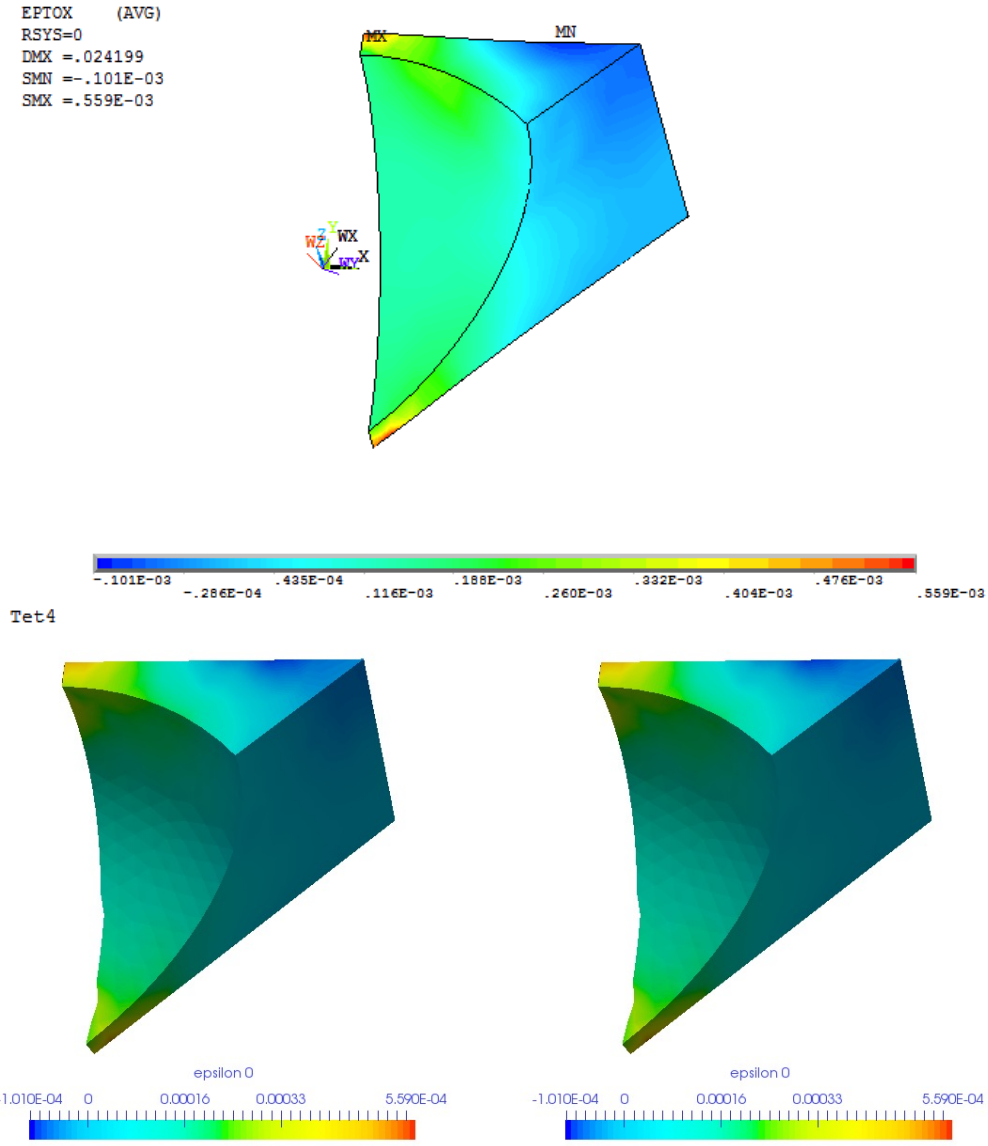

Figure 4.39: Mesh with Tet4, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

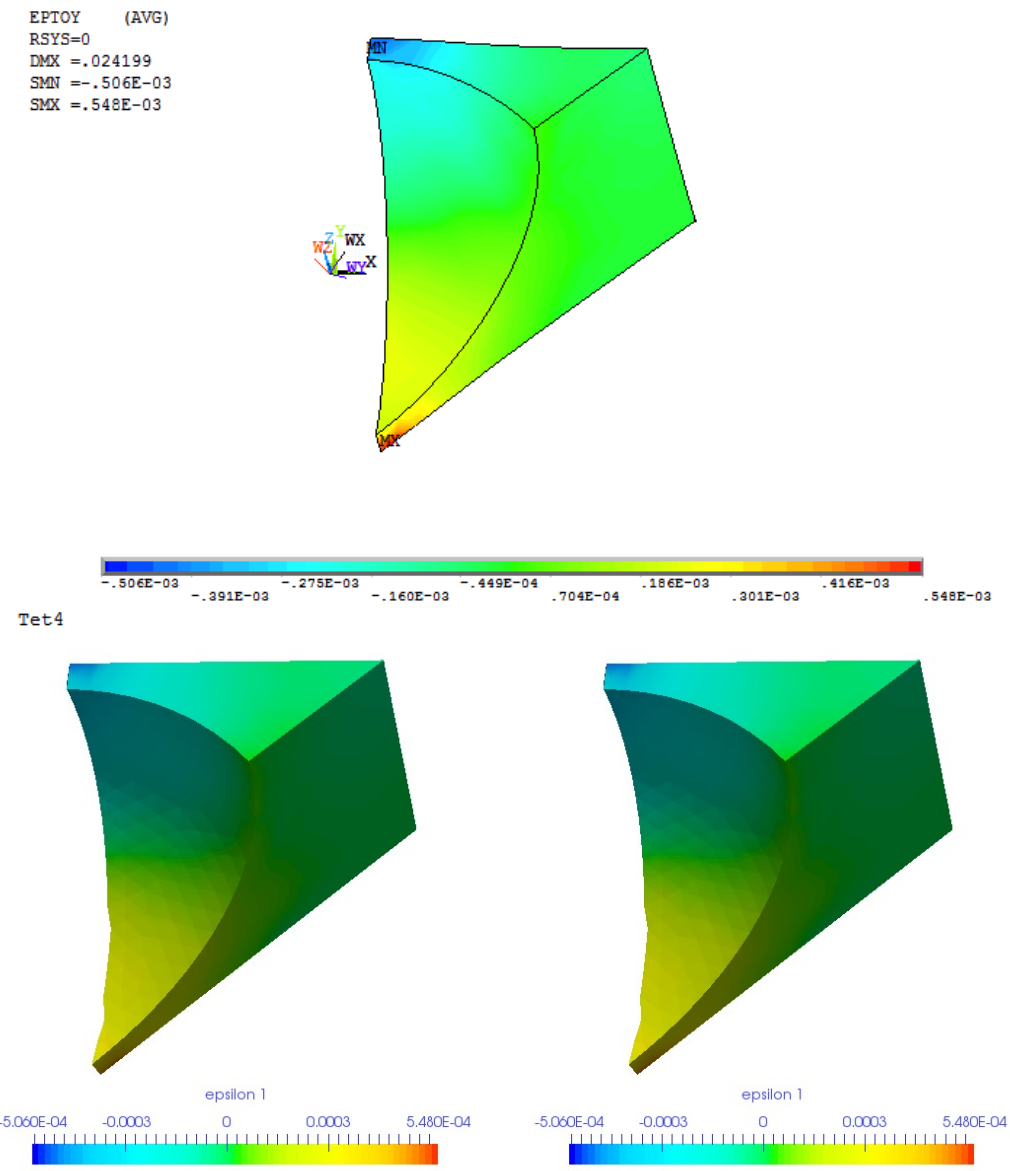

Figure 4.40: Mesh with Tet4, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

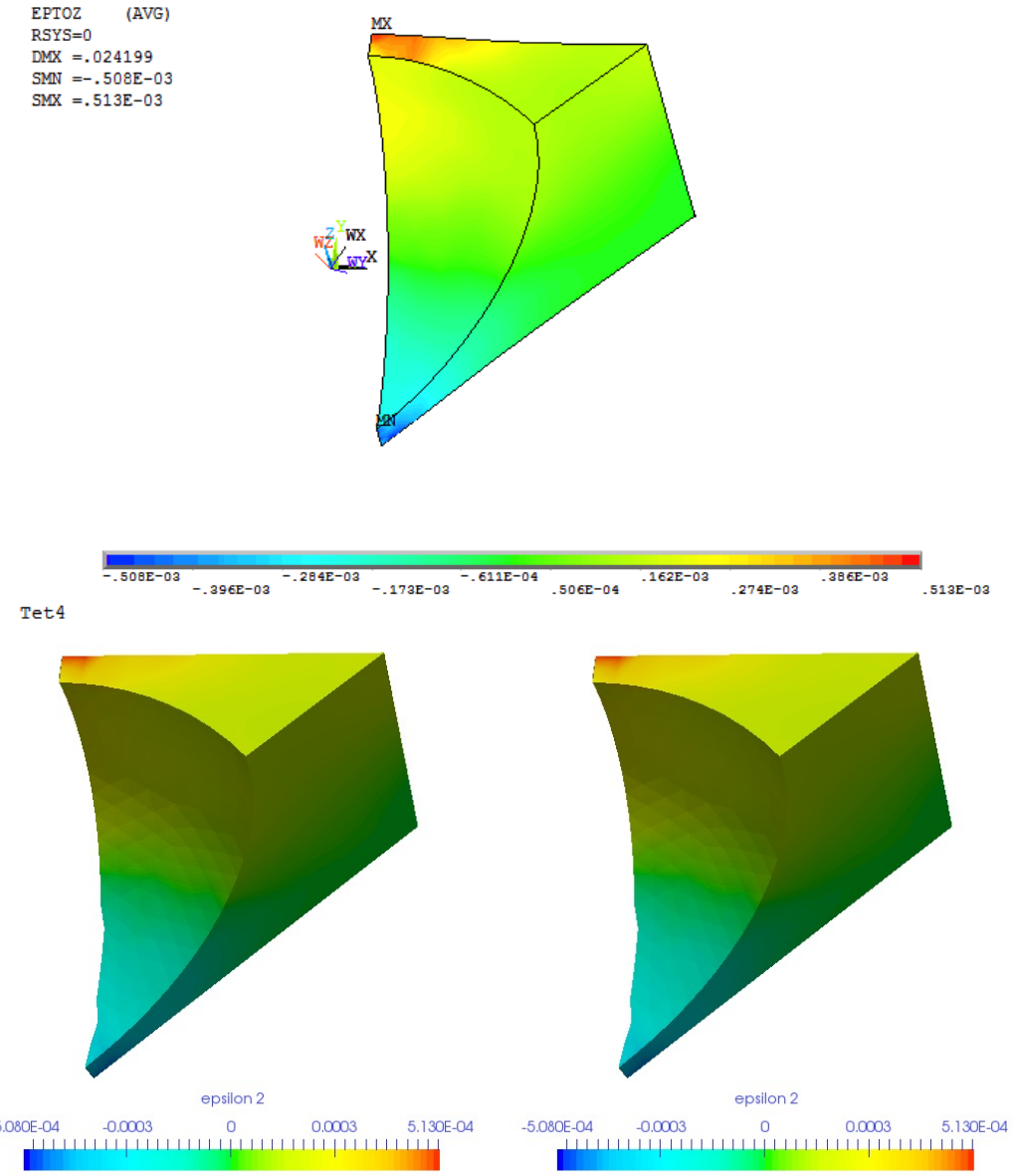

Figure 4.41: Mesh with Tet4, upper: contour plot of ϵ_{zz} in ANSYS; lower left: contour plot of ϵ_{zz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{zz} calculated with stress recovery in AMfe

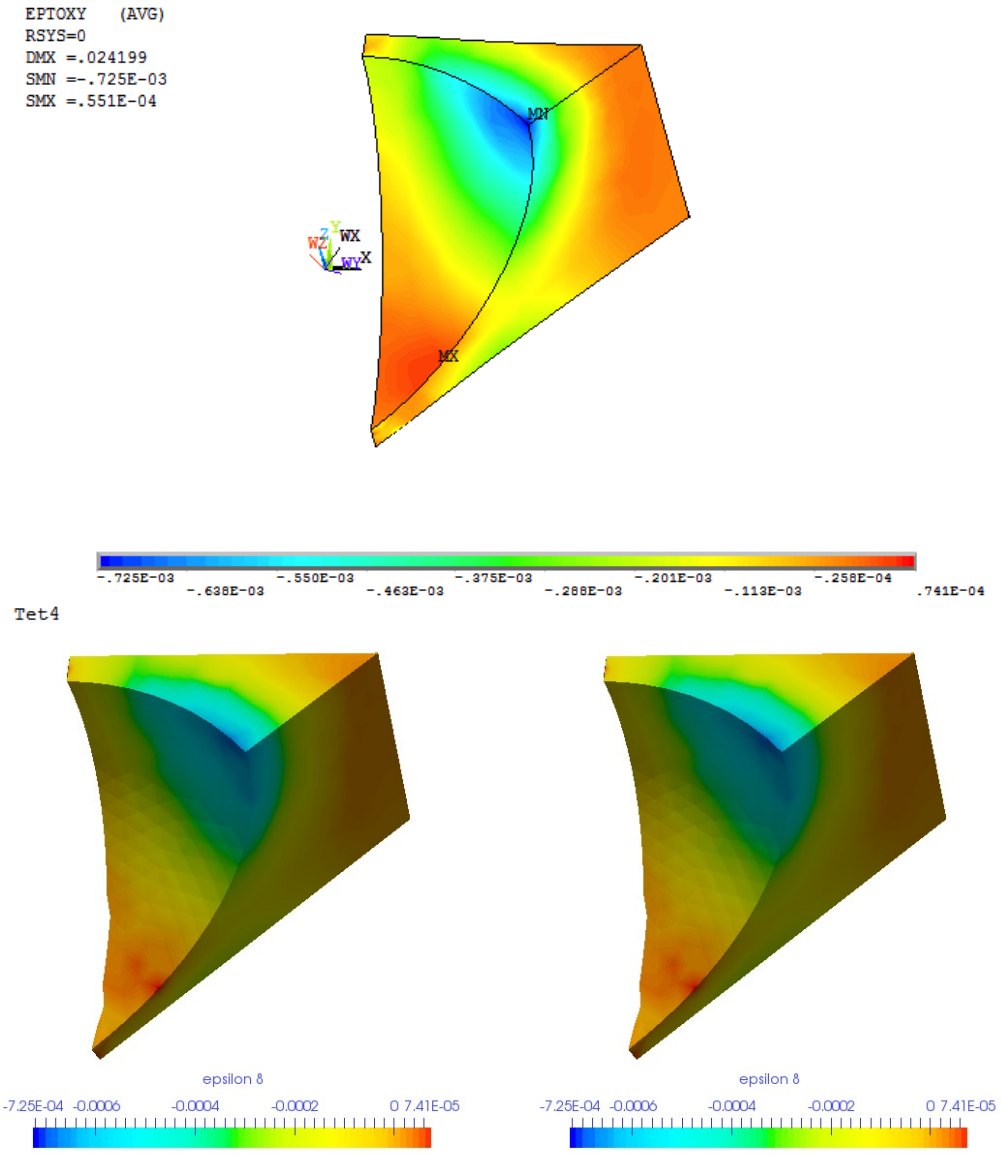

Figure 4.42: Mesh with Tet4, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

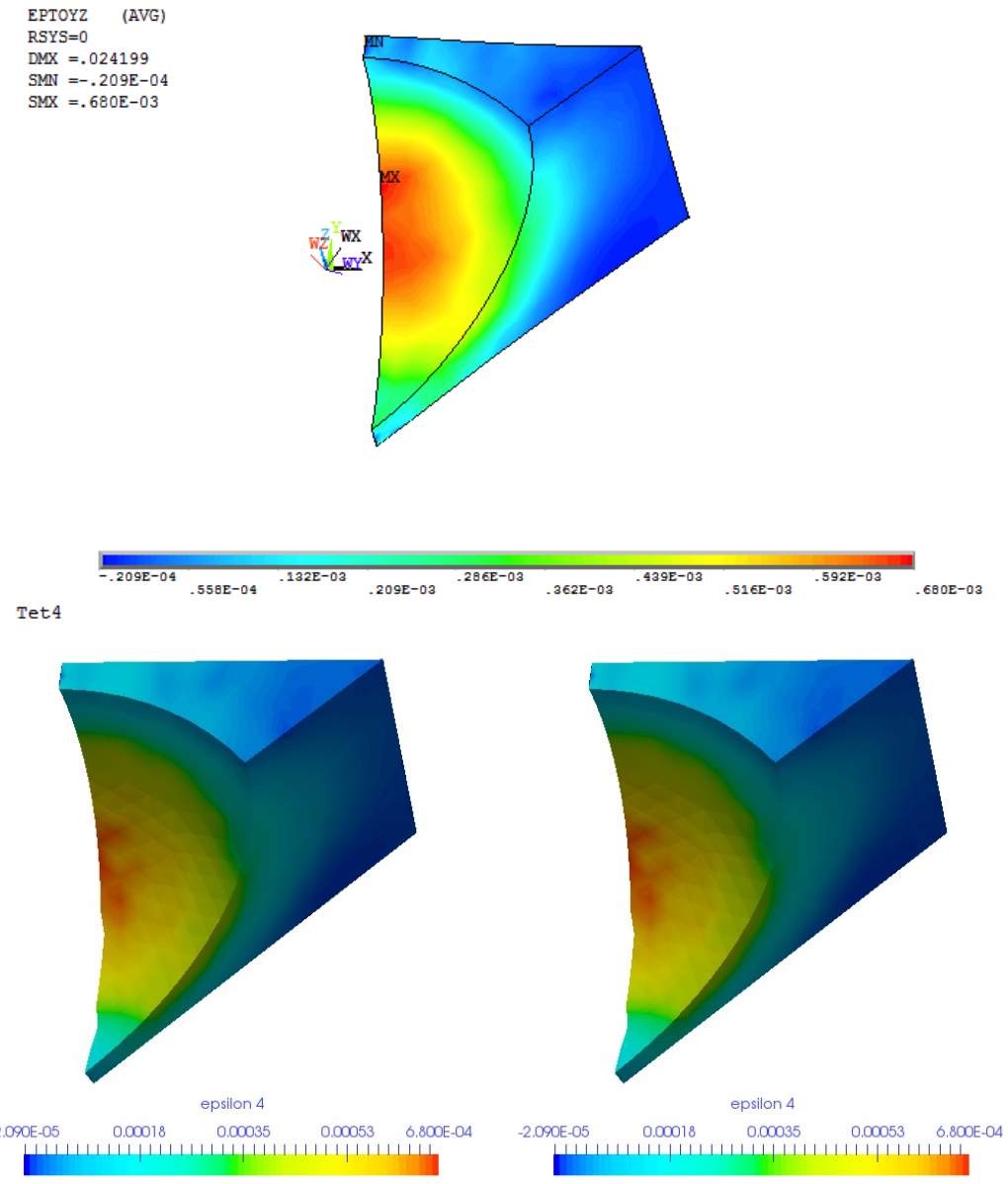

Figure 4.43: Mesh with Tet4, upper: contour plot of ϵ_{yz} in ANSYS; lower left: contour plot of ϵ_{yz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yz} calculated with stress recovery in AMfe

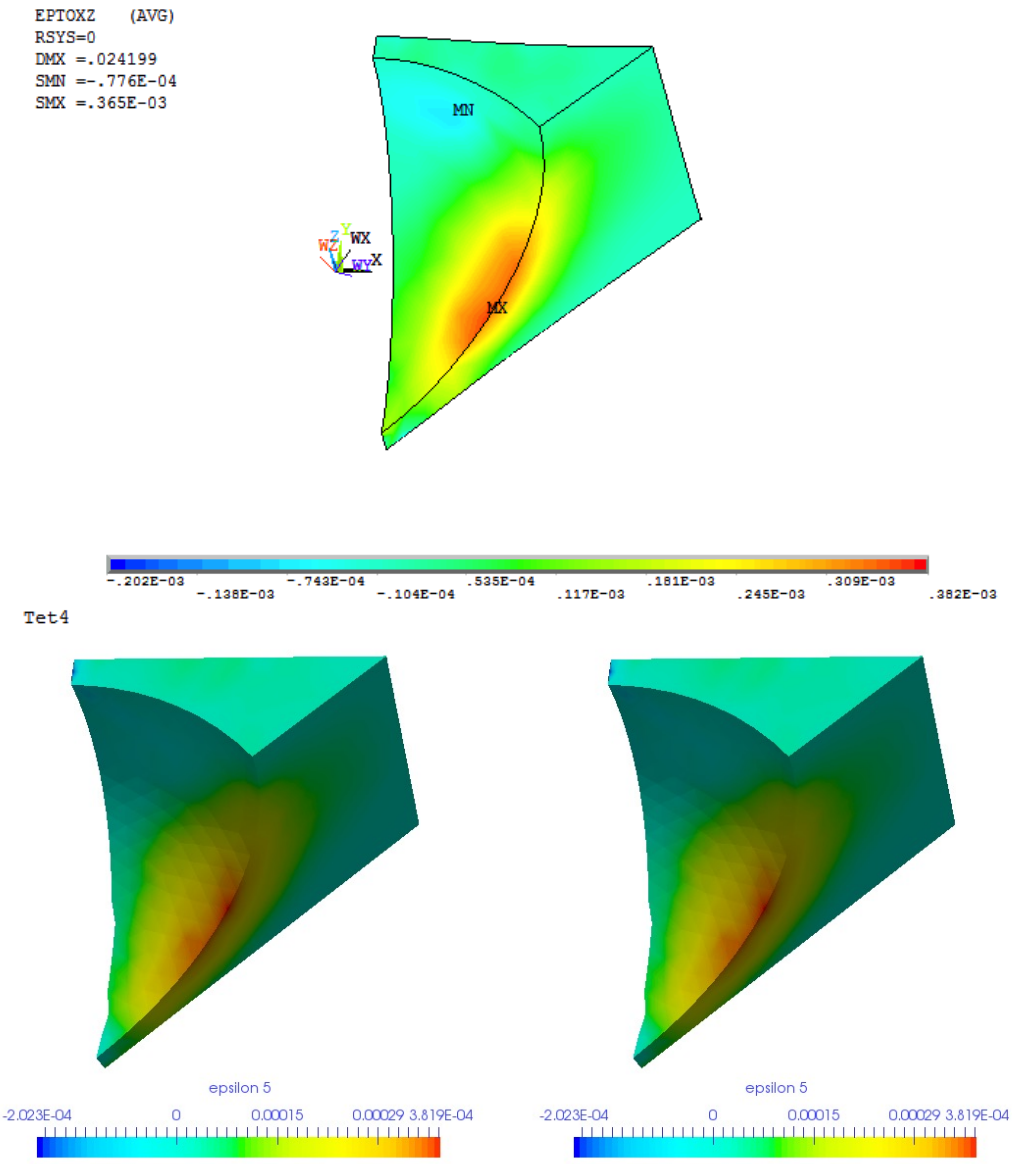

Figure 4.44: Mesh with Tet4, upper: contour plot of ϵ_{xz} in ANSYS; lower left: contour plot of ϵ_{xz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xz} calculated with stress recovery in AMfe

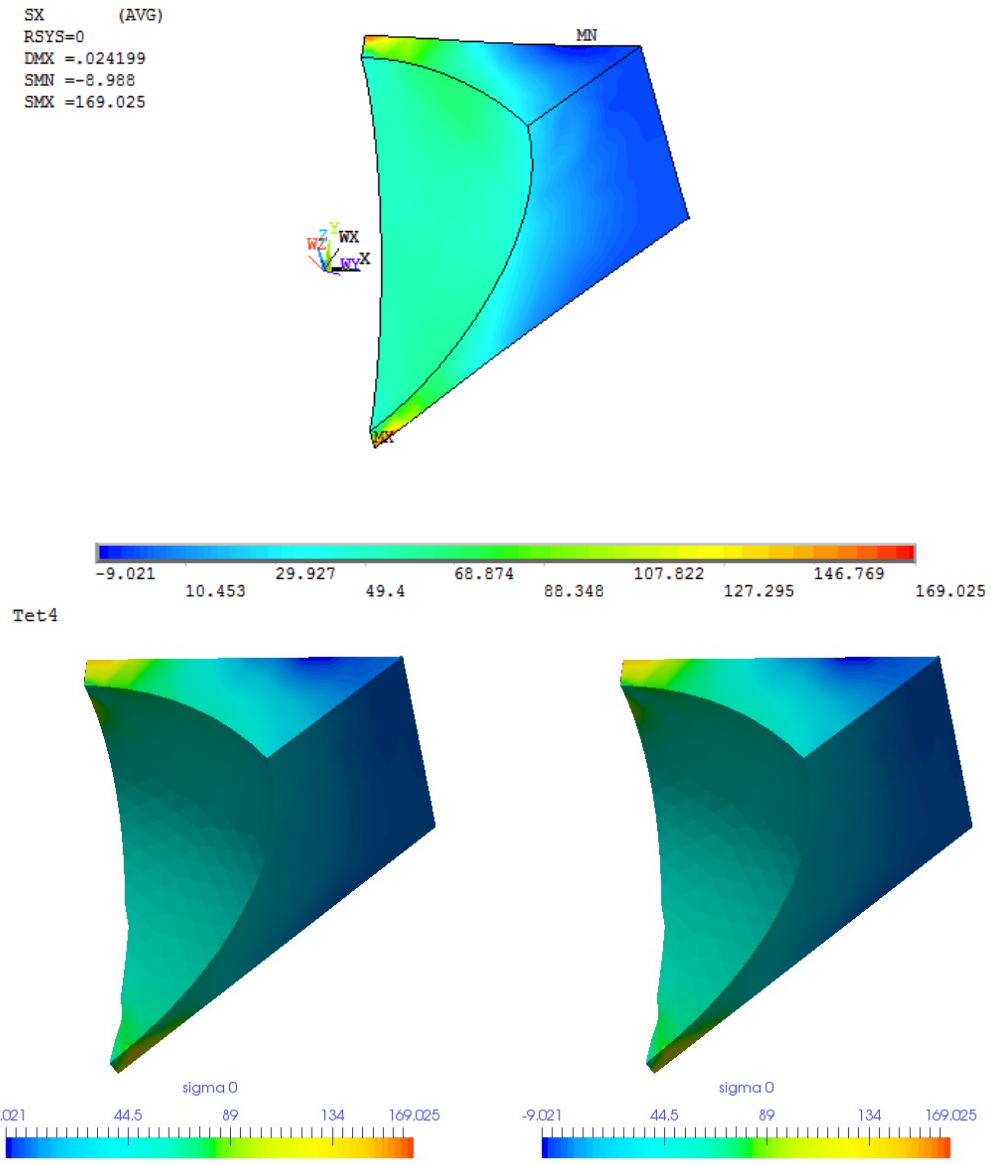

Figure 4.45: Mesh with Tet4, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

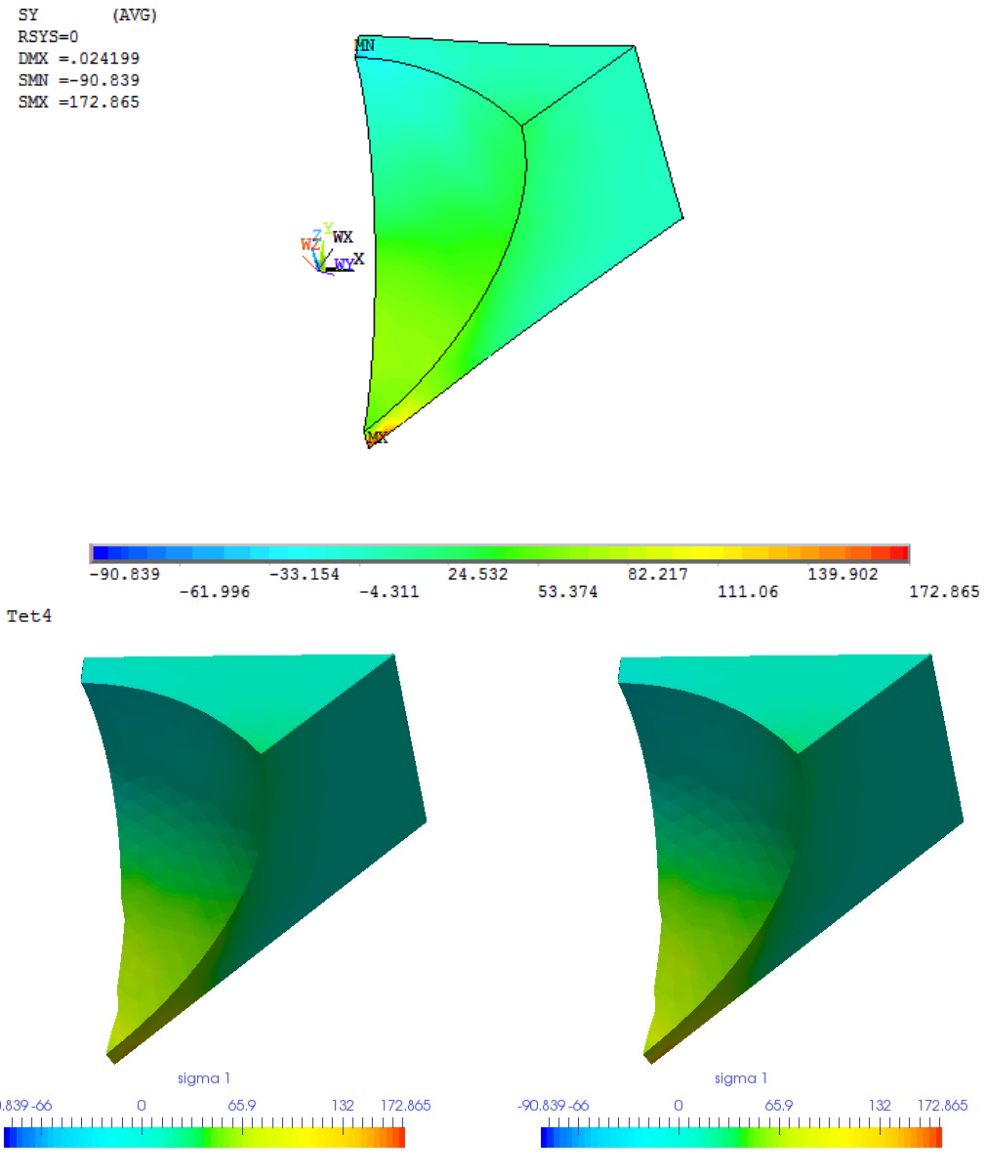

Figure 4.46: Mesh with Tet4, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

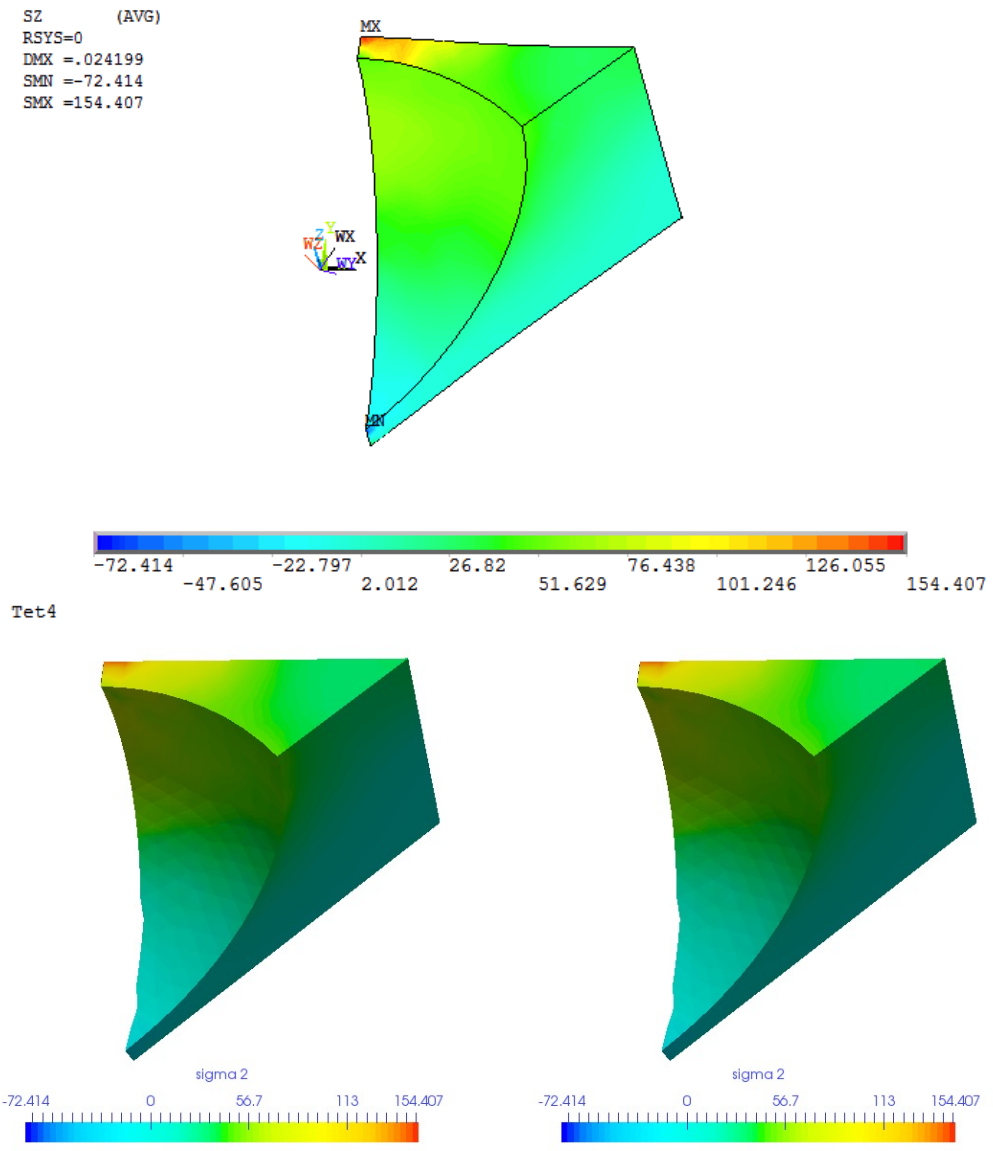

Figure 4.47: Mesh with Tet4, upper: contour plot of σ_{zz} in ANSYS; lower left: contour plot of σ_{zz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{zz} calculated with stress recovery in AMfe

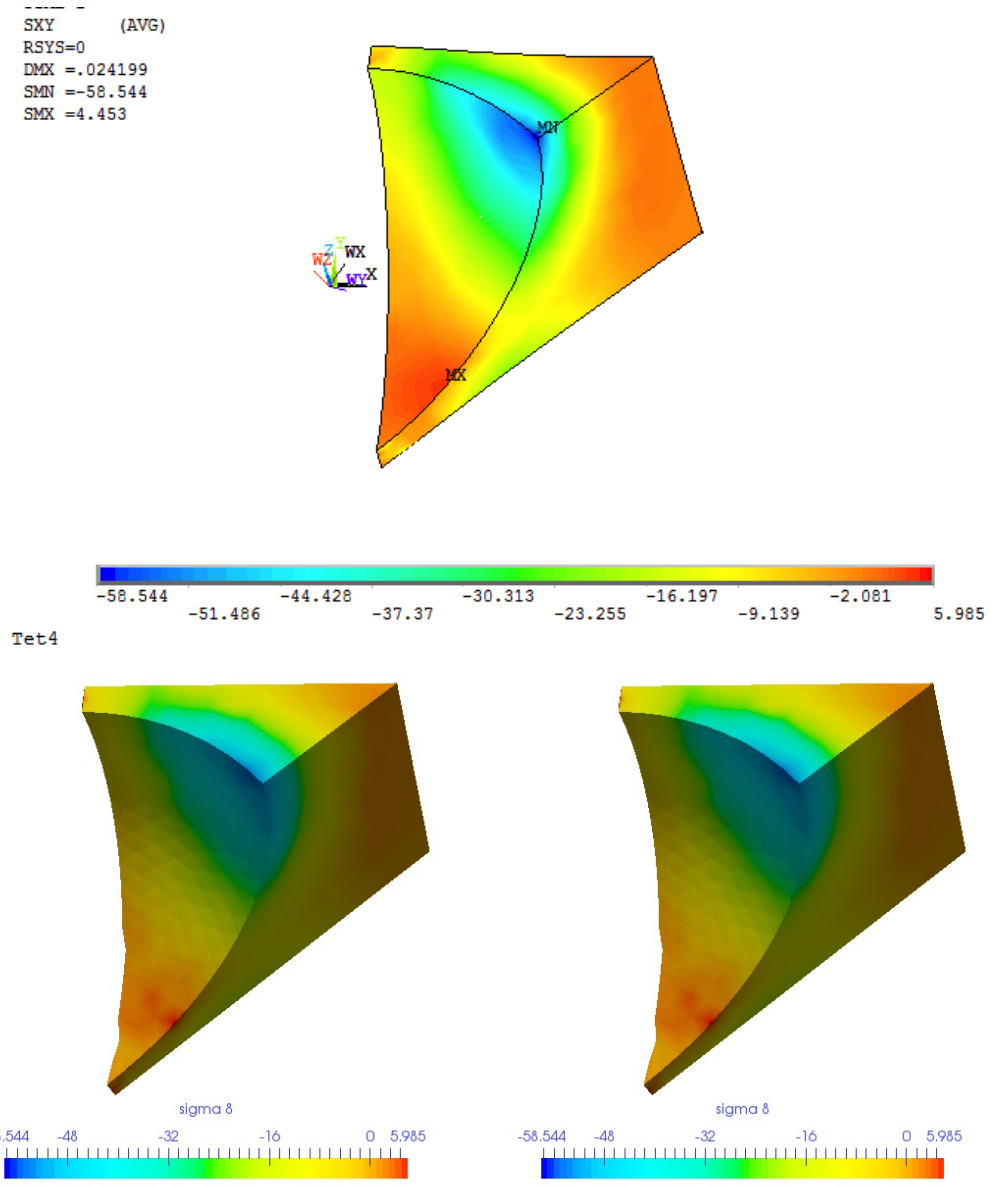

Figure 4.48: Mesh with Tet4, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

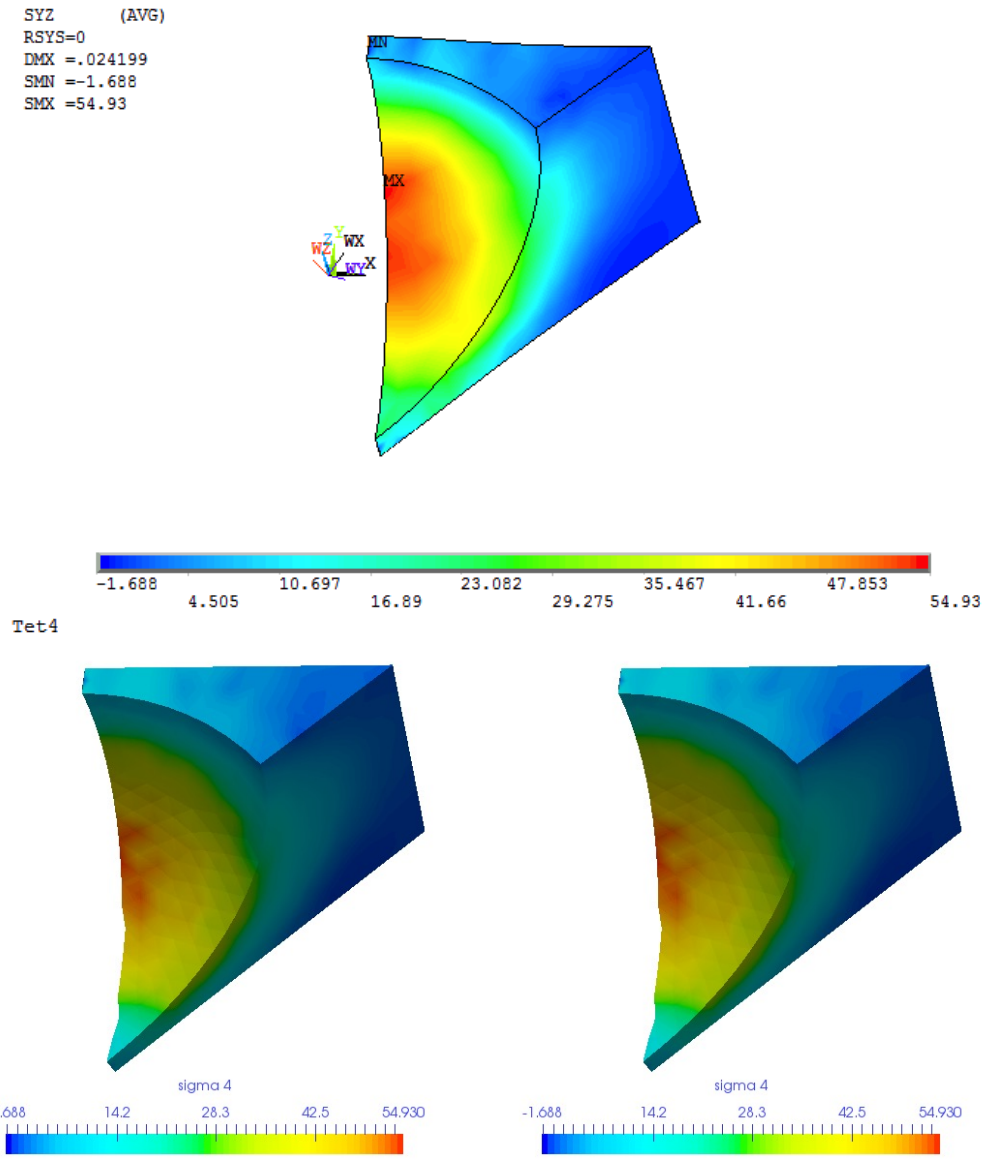

Figure 4.49: Mesh with Tet4, upper: contour plot of σ_{yz} in ANSYS; lower left: contour plot of σ_{yz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yz} calculated with stress recovery in AMfe

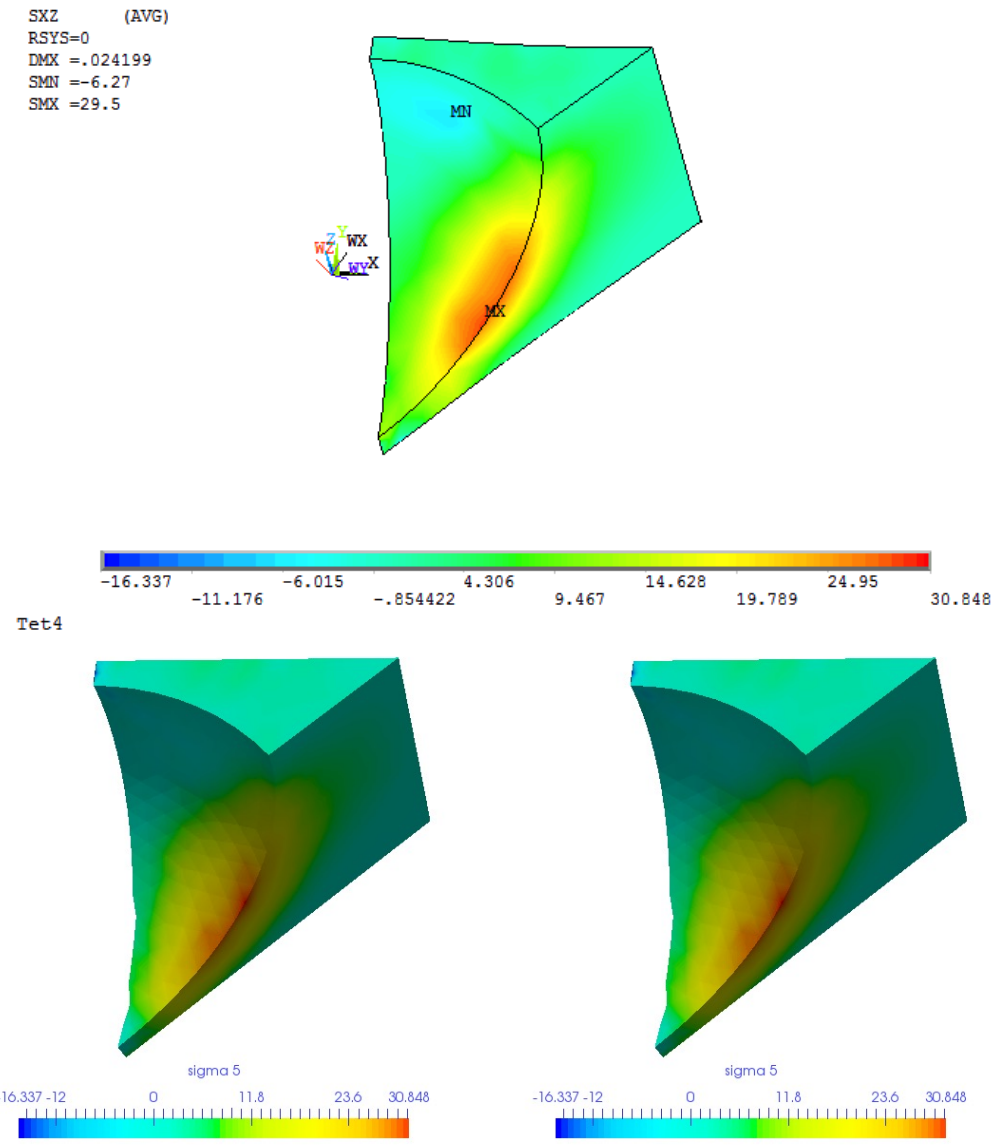

Figure 4.50: Mesh with Tet4, upper: contour plot of σ_{xz} in ANSYS; lower left: contour plot of σ_{xz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xz} calculated with stress recovery in AMfe

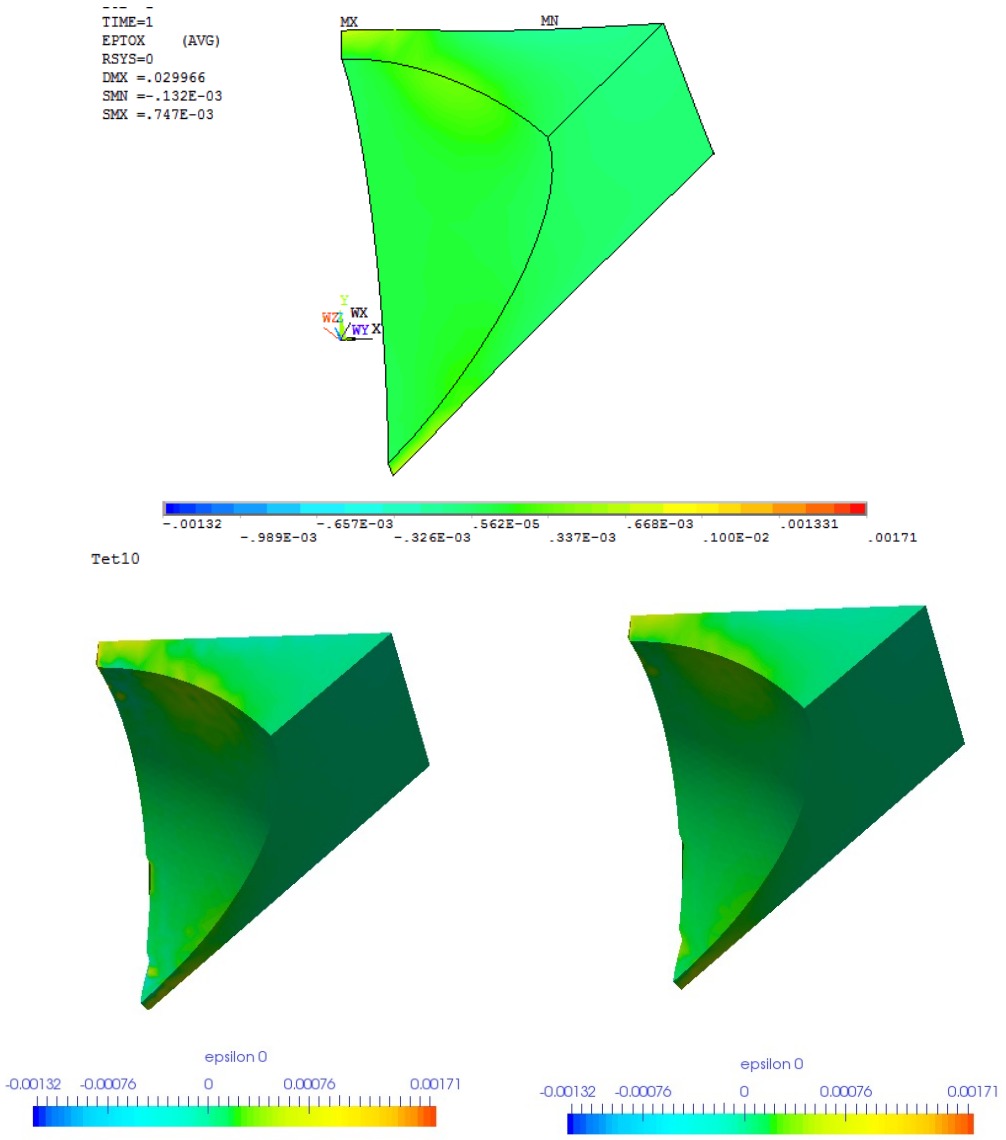

Figure 4.51: Mesh with Tet10, upper: contour plot of ϵ_{xx} in ANSYS; lower left: contour plot of ϵ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xx} calculated with stress recovery in AMfe

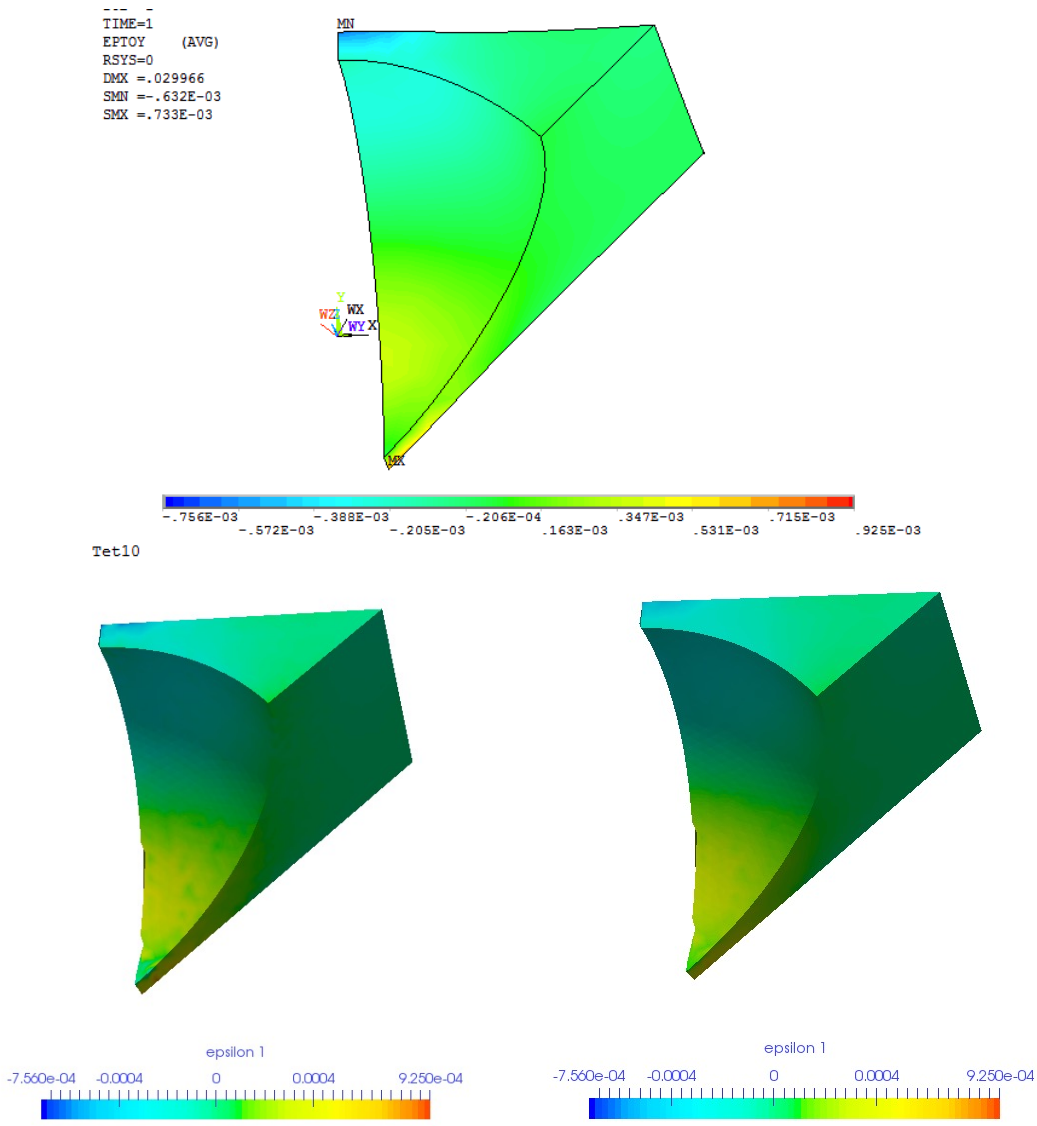

Figure 4.52: Mesh with Tet10, upper: contour plot of ϵ_{yy} in ANSYS; lower left: contour plot of ϵ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yy} calculated with stress recovery in AMfe

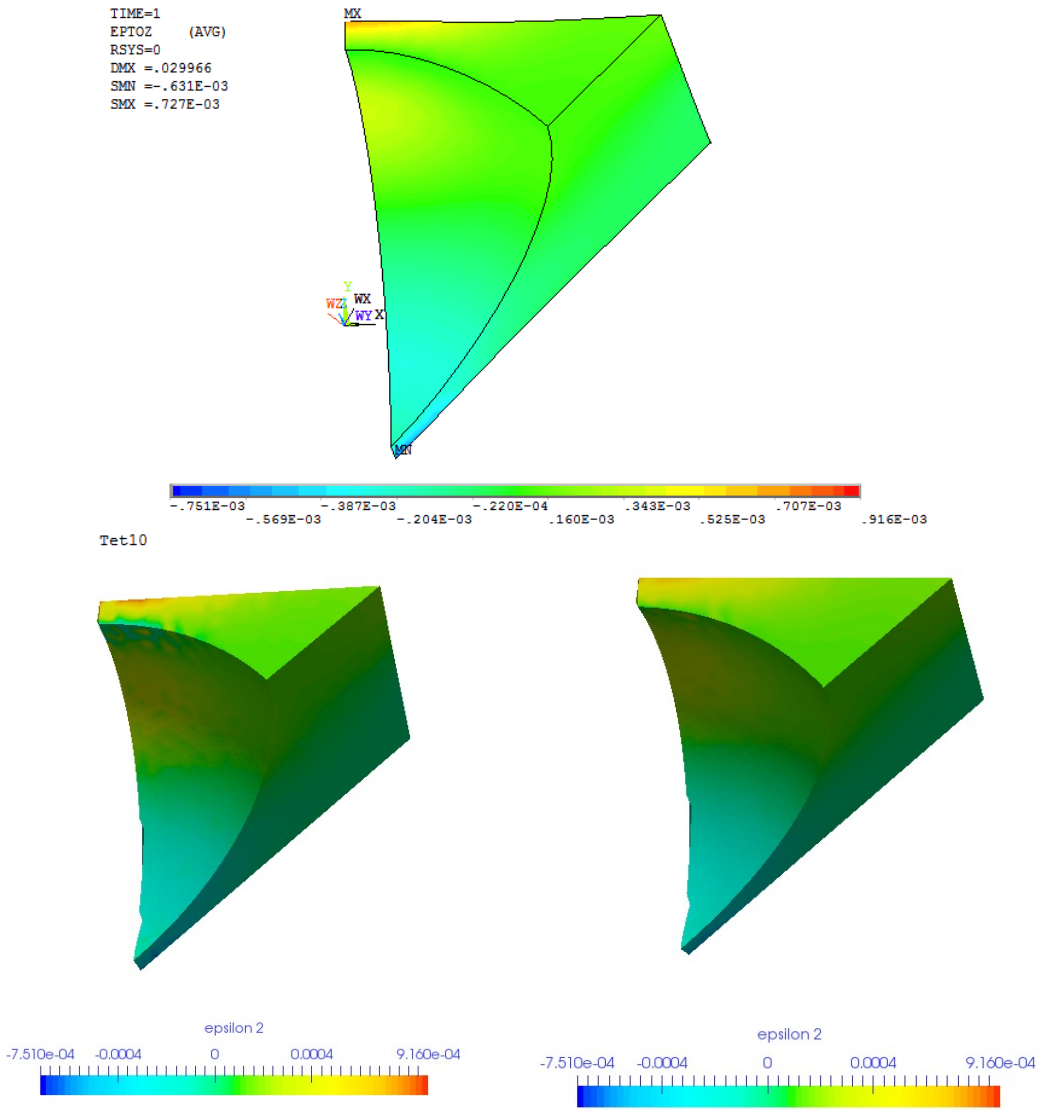

Figure 4.53: Mesh with Tet10, upper: contour plot of ϵ_{zz} in ANSYS; lower left: contour plot of ϵ_{zz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{zz} calculated with stress recovery in AMfe

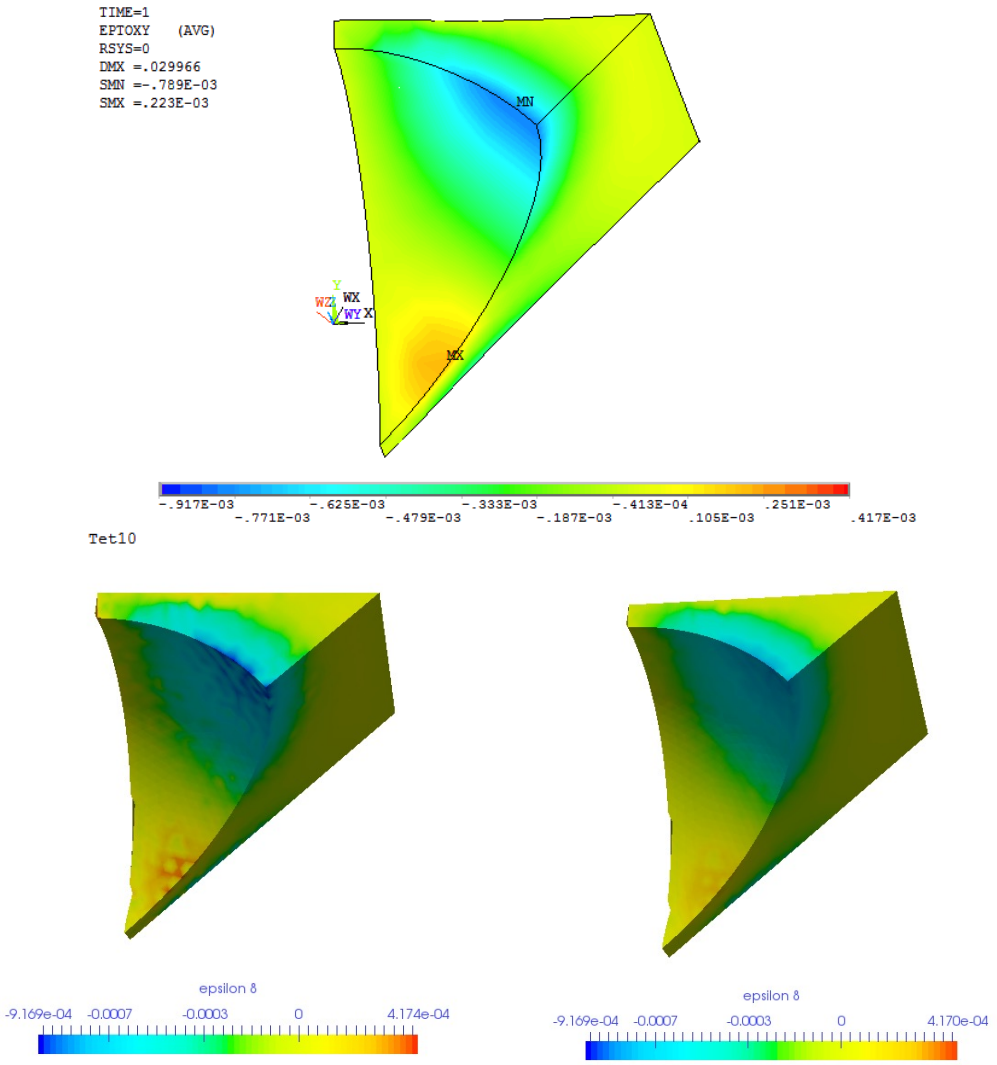

Figure 4.54: Mesh with Tet10, upper: contour plot of ϵ_{xy} in ANSYS; lower left: contour plot of ϵ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xy} calculated with stress recovery in AMfe

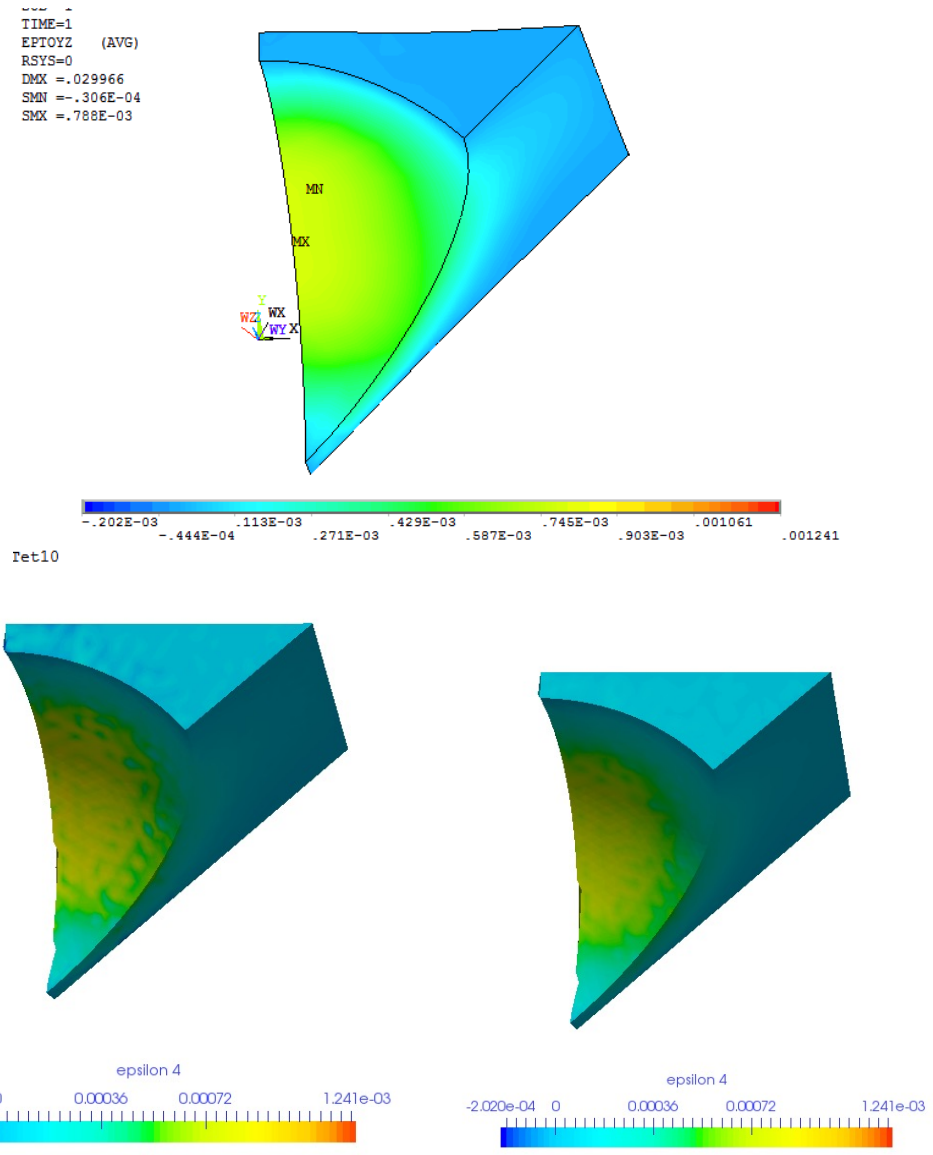

Figure 4.55: Mesh with Tet10, upper: contour plot of ϵ_{yz} in ANSYS; lower left: contour plot of ϵ_{yz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{yz} calculated with stress recovery in AMfe

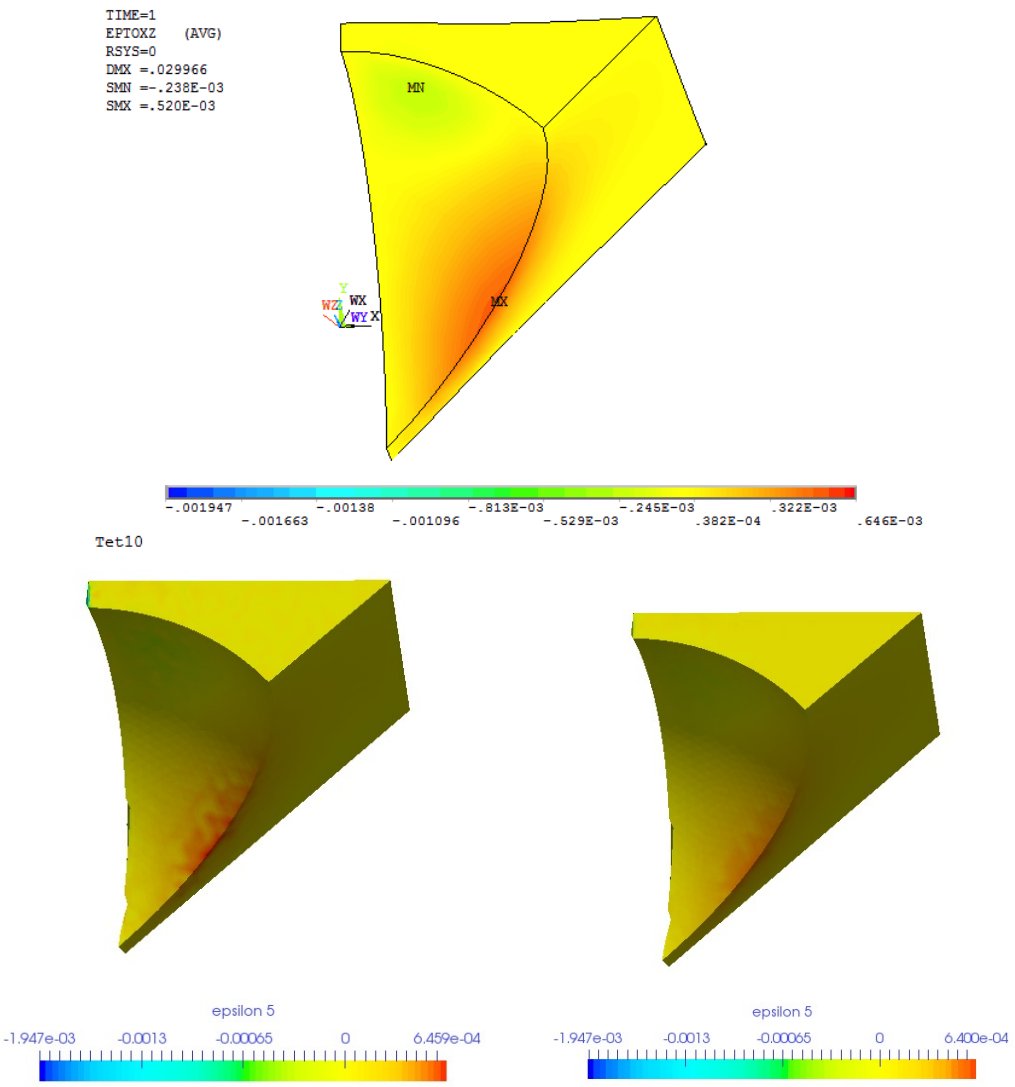

Figure 4.56: Mesh with Tet10, upper: contour plot of ϵ_{xz} in ANSYS; lower left: contour plot of ϵ_{xz} calculated without stress recovery in AMfe; lower right: contour plot of ϵ_{xz} calculated with stress recovery in AMfe

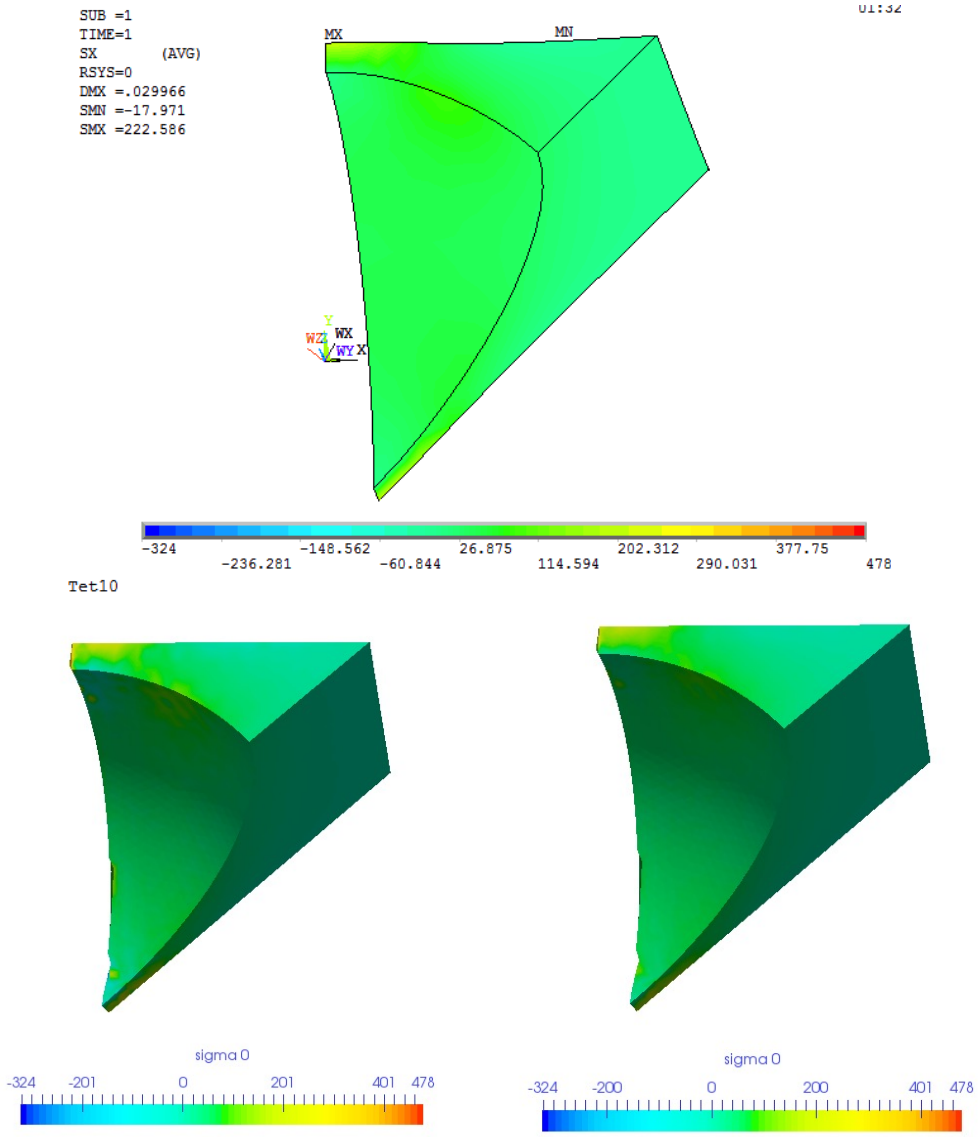

Figure 4.57: Mesh with Tet10, upper: contour plot of σ_{xx} in ANSYS; lower left: contour plot of σ_{xx} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xx} calculated with stress recovery in AMfe

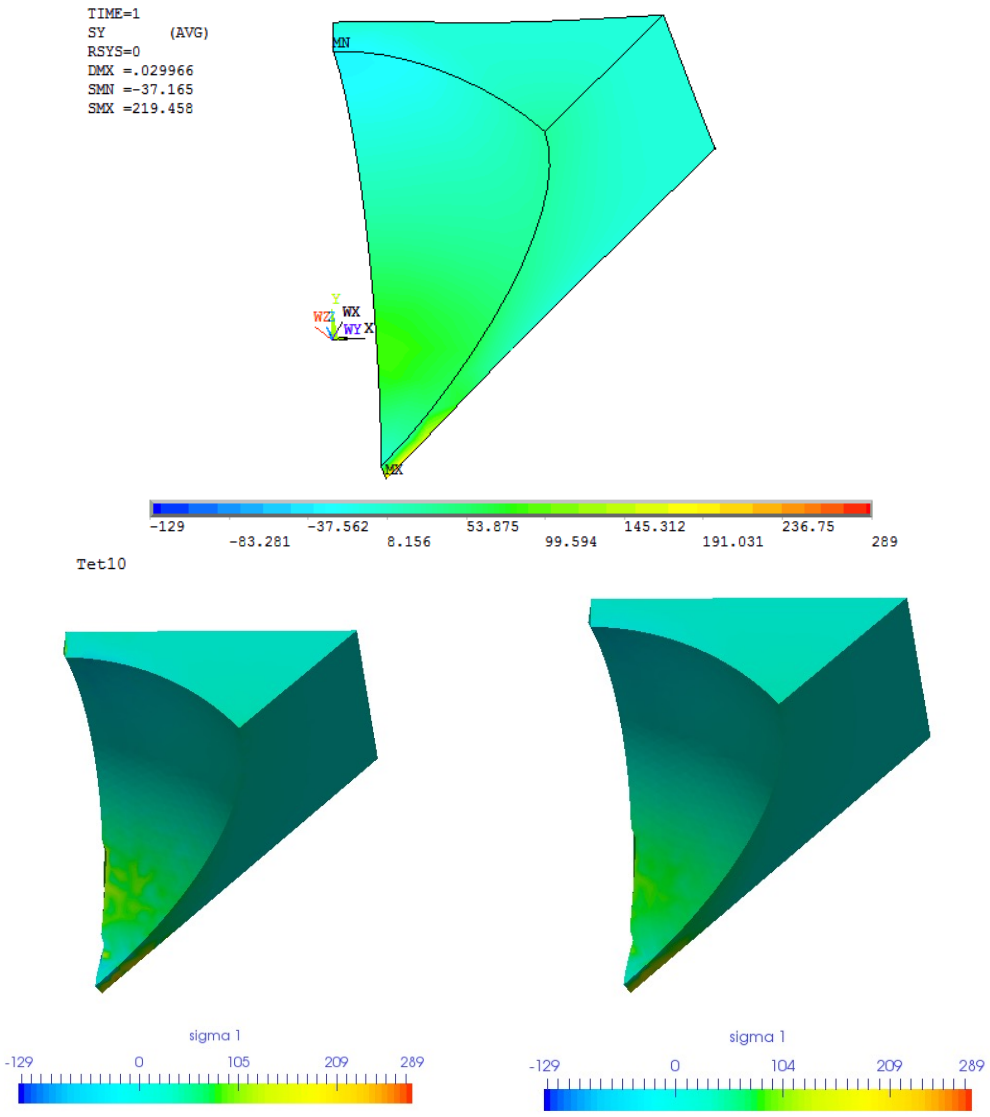

Figure 4.58: Mesh with Tet10, upper: contour plot of σ_{yy} in ANSYS; lower left: contour plot of σ_{yy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yy} calculated with stress recovery in AMfe

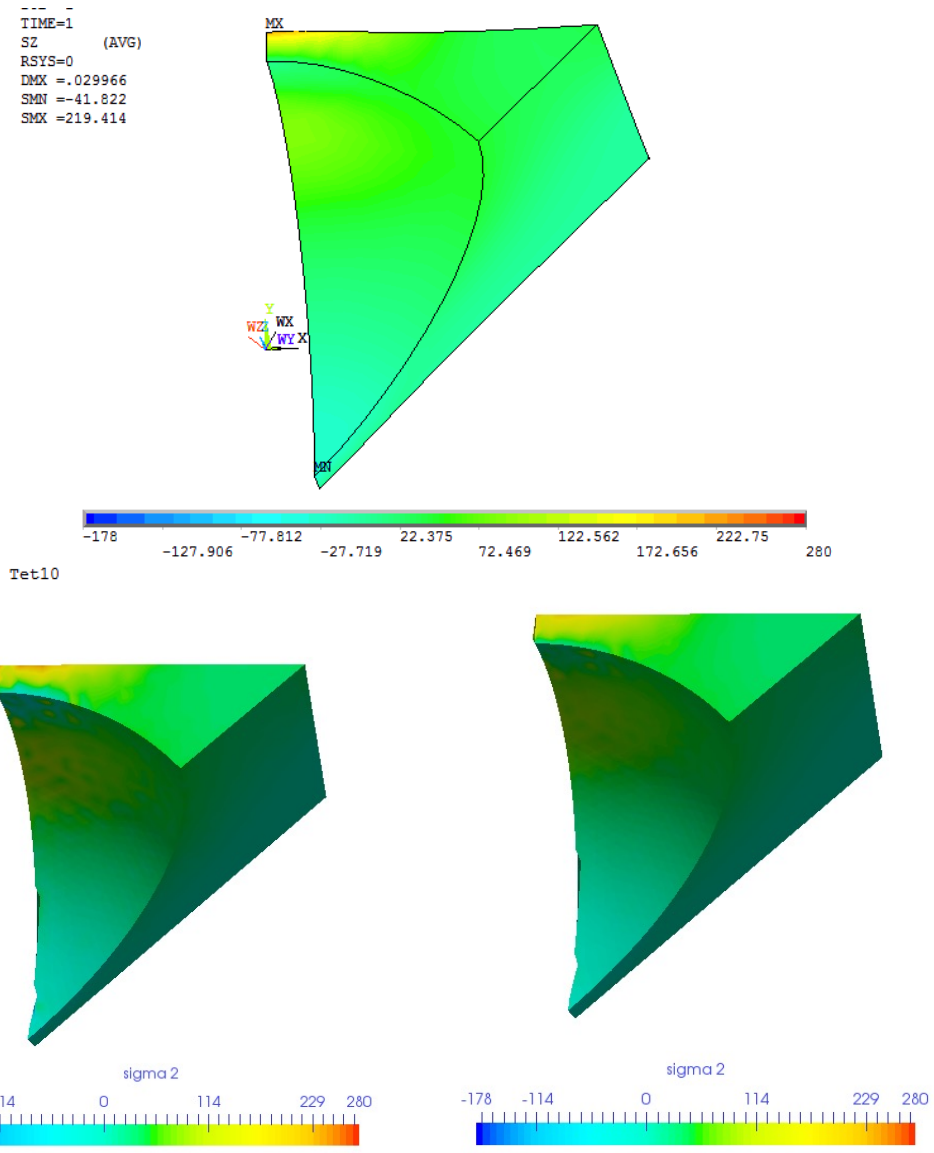

Figure 4.59: Mesh with Tet10, upper: contour plot of σ_{zz} in ANSYS; lower left: contour plot of σ_{zz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{zz} calculated with stress recovery in AMfe

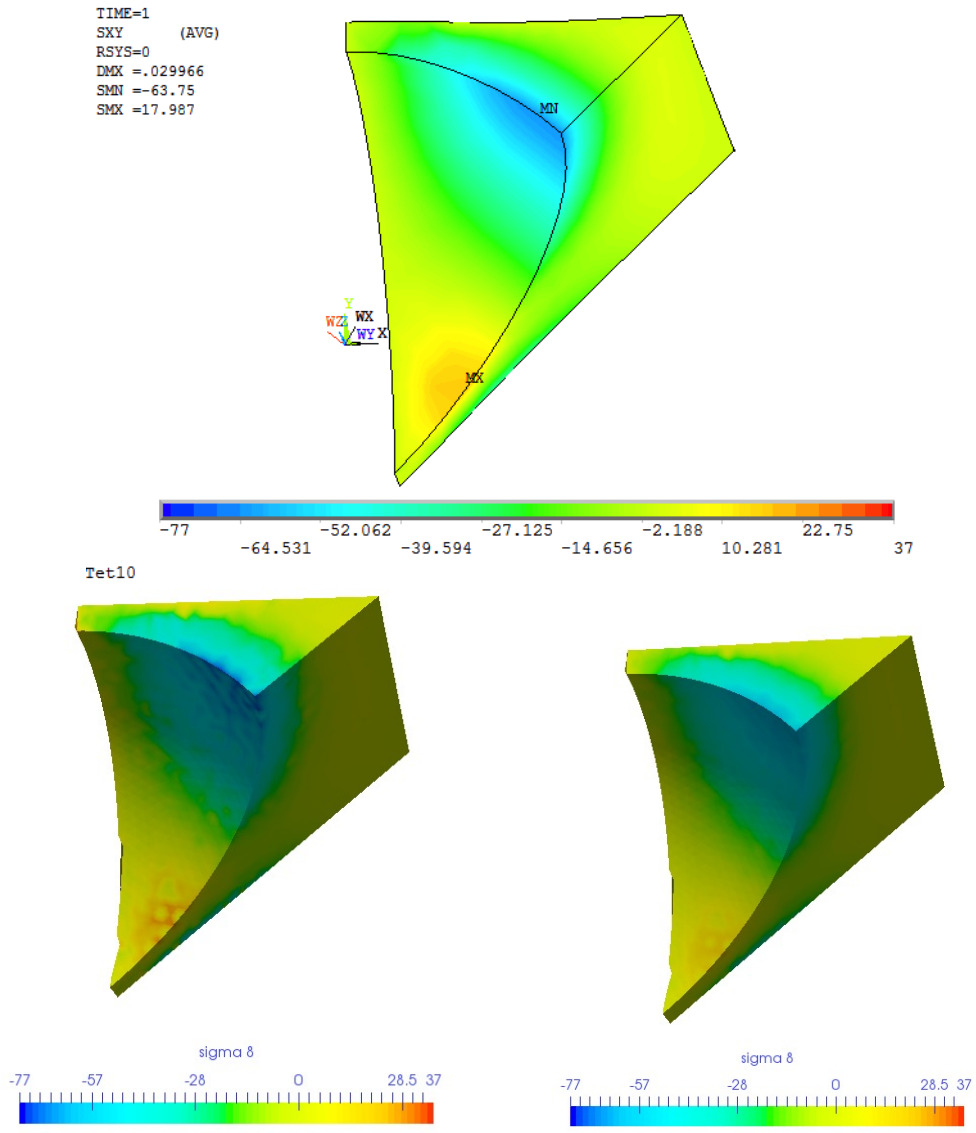

Figure 4.60: Mesh with Tet10, upper: contour plot of σ_{xy} in ANSYS; lower left: contour plot of σ_{xy} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xy} calculated with stress recovery in AMfe

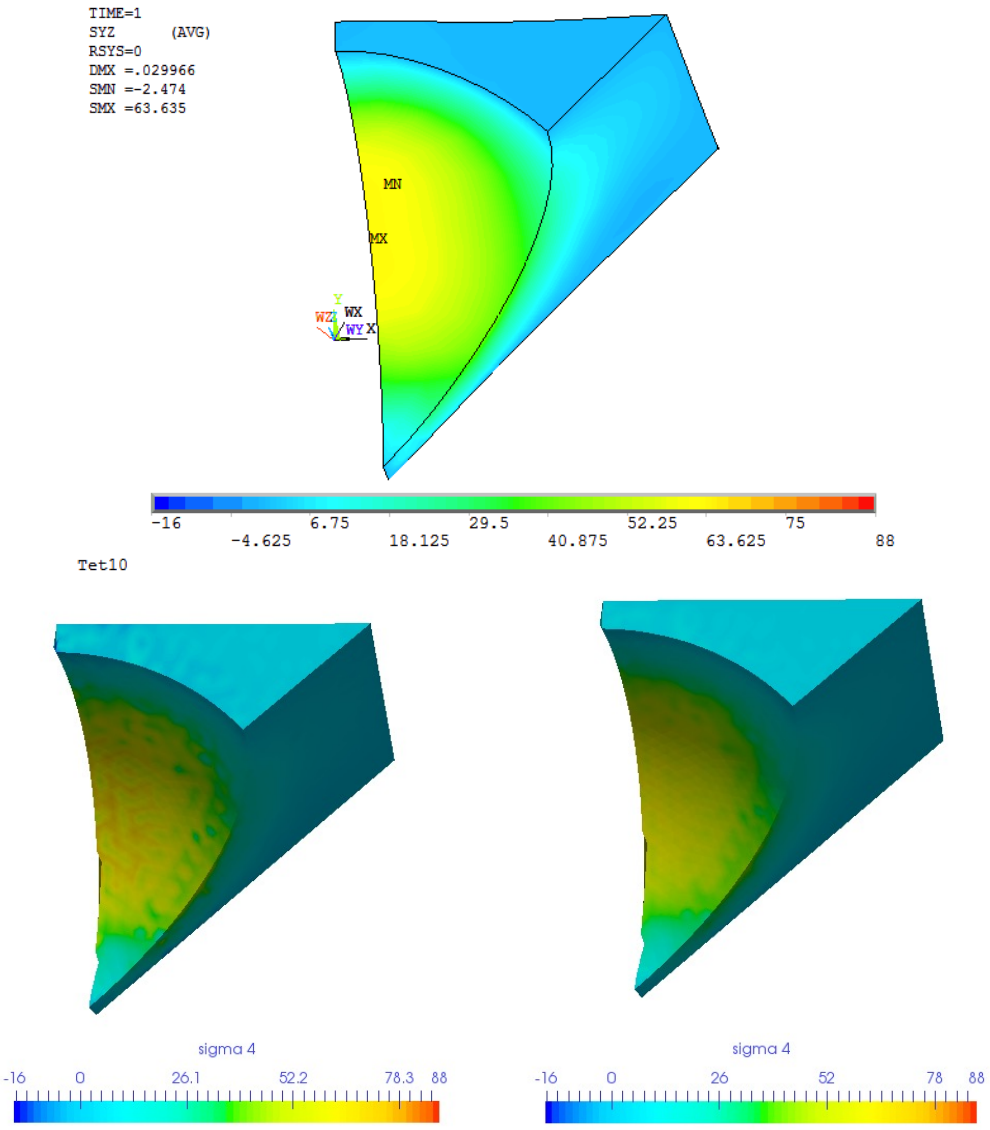

Figure 4.61: Mesh with Tet10, upper: contour plot of σ_{yz} in ANSYS; lower left: contour plot of σ_{yz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{yz} calculated with stress recovery in AMfe

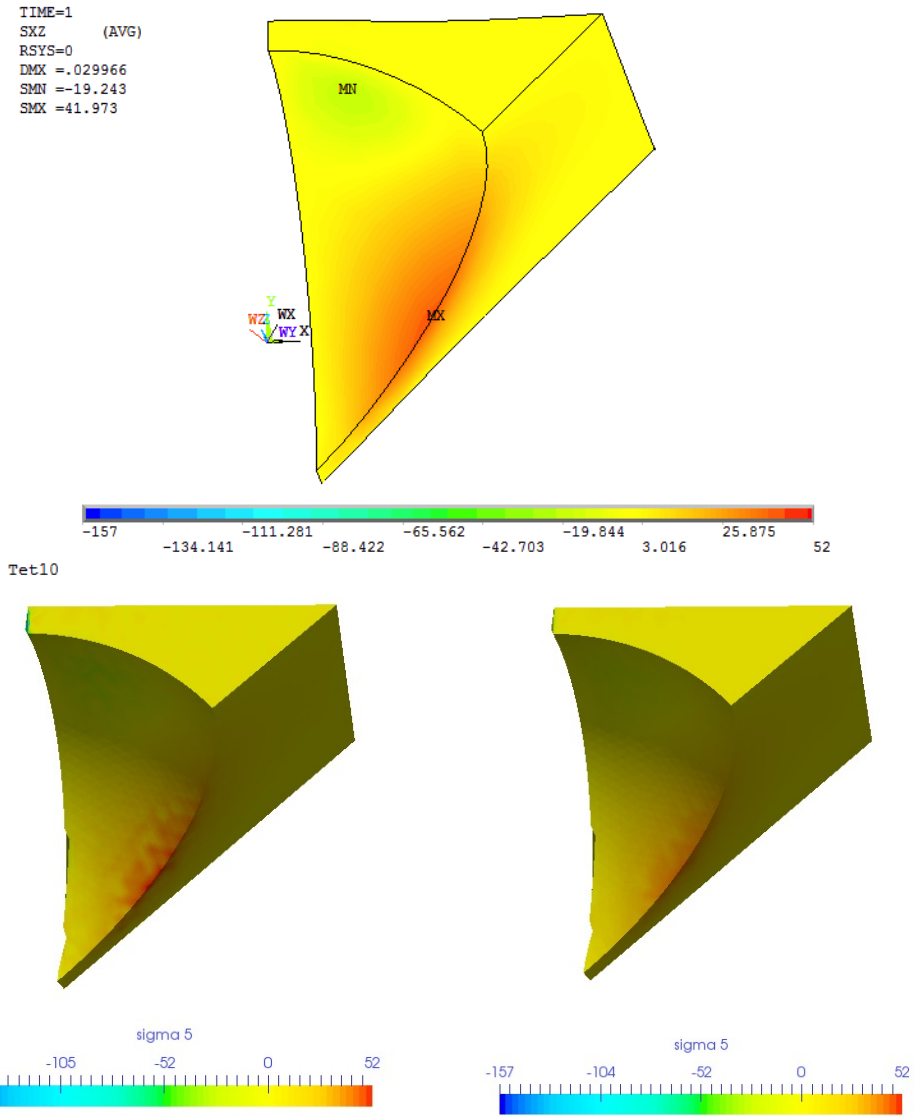

Figure 4.62: Mesh with Tet10, upper: contour plot of σ_{xz} in ANSYS; lower left: contour plot of σ_{xz} calculated without stress recovery in AMfe; lower right: contour plot of σ_{xz} calculated with stress recovery in AMfe

Chapter 5

Conclusion

5.1 Summary

There are two algorithms to get the approximation of nodal stress, one is Direct-Evaluation-Algorithms and the other one is Stress-Recovery-Algorithms. The latter one is based on an extrapolation from the Gauss points to the corner nodes. According to different element types, it is necessary to choose the suitable coordinate systems. For quadrilateral elements, the Cartesian coordinate system is appropriate to choose. For triangular and tetrahedral elements, their own coordinate system is more appropriate for the calculation. The type of Gauss element for each element types is also available for selection. We take the selection as follows: 2×2 quadrilateral element with four Gauss points for Quad4; 3×3 quadrilateral element with nine Gauss points for Quad8; triangular element with three Gauss points for Tri3; triangular element with six Gauss points for Tri6; tetrahedral element with four Gauss points for Tet4 and Tet10. The step after extrapolation of stress from Gauss points is to solve the average value at each node by dividing the number of elements that share the same node. And then the nodal stress is finally 'recovered'.

In the implementation part of this thesis, we export the Pre-Processing data from ANSYS modeling and import into AMfe Toolbox for calculation. Then the solutions by Stress-Recovery-Algorithms and Direct-Evaluation-Algorithms are compared to the ANSYS solution. An unit testing for calculating the Von Mises stress shows that only Tri3 and Quad4 have an acceptable accuracy of Von Mises stress. Thus we do the displacement patch test for each element by the both algorithms. This results of the patch test show that all the element types by both algorithms have passed the patch test and Tri6, Quad8, Tet4, and Tet10 get the constant strain value as 1.5, which expect to 1. And the reason for this issue has been found out by debugging the code. The Pre-calculated strain at Gauss points is 1.5, when the modelling are meshed with Tri6, Quad8, Tet4, and Tet10. Hence the nodal strain after extrapolation is also 1.5. This proves that the Stress-Recovery-Algorithms works well. In addition, the convergence analysis for displacement and Von Mises stress are also checked. The results of displacements convergence analysis represents that all the element types converges to the exact solution. The higher order elements (Tri6 and Tet10) have a relatively higher convergence rate than the lower order elements (Tri3 and Tet4) and this characteristic is not that discernible with quadrilateral elements. The results of Von Mises stress convergence analysis represents that all the element types converges to the exact

solution and the elements with higher order (Tri6, Quad8, and Tet10) converges faster than the one with lower order (Tri3, Quad4, and Tet4). The results also show that the Quad4 has a higher convergence rate than Tri3. The last test is to observe the performance of all the element types in a relative complex model. We can not only compare the strain/stress contour plots between the solution by Direct-Evaluation-Algorithms and the solution by Stress-Recovery-Algorithms, but also the strain/stress contour plots between the solution from ANSYS and the solution from AMfe by the both algorithms. The comparison among these three solution express that there is no difference between the solution by the both algorithms for Tri3 and Tet4, and the solution by Stress-Recovery-Algotithms has a better performance than the solution by Direct-Evaluation-Algorithms for the other element types (Tri6, Quad4, Quad8, and Tet10). The issue by Direct-Evaluation-Algorithms is the deviation at the value around the extreme values, namely the value around maximum and minimum value. The values at the middle part appear rather closer to the ANSYS solution. In summary, Stress-Recovery-Algorithms improves the accuracy of nodal stress for Tri6, Quad4, Quad8, and Tet10 with respect to the Direct-Evaluation-Algorithms.

5.2 Recommendation

According to the work of Carlos A. Felippa [Felippa 2004 section 28 p.6], there are two method to calculate the averaged nodal stress after stress recovery. One is the approach, which we apply in this paper, unweighted averaging for the nodal stress. The other one is weighted averaging for the nodal stress, which allocates the weights to the contributions of element sharing the same node. Stress components, element types could affect the weights for selection.

According to the patch test in this paper, we find the issue with the strain at Gauss points with element types Tri6, Quad8, Tet4, and Tet10. Therefore, an investigation for the calculation of strain at Gauss points with these mentioned problematic element types is meaningful to improve the accuracy of nodal strain and stress.

We discuss in this thesis only about the case of extrapolation from one type of Gauss element. A comparison and discuss of the nodal stress between the extrapolation by using different types of Gauss element would be helpful for the further Stress-Recovery-Algorithms.

qwe Wang Felippa 2004**qwe**

Bibliography

Felippa, Carlos A. (2004). *INTRODUCTION to Introduction to Finite Element Methods*. Boulder, Colorado 80309-0429, USA: University of Colorado.

Dukkipati, Rao V. (2010). *Numerical Methods*. 4835/24 Ansari Road, Daryaganj, New Delhi: New Age International Publishers.

ANSYS, Inc (2011). *ANSYS Parametric Design Language Guide*. Release 14.0. ANSYS, Inc. Canonsburg, PA 15317.

Kurowski, P. M. (2012). *Von Mises Stress failure criterion*.

Wall, W. A. (2014). *Finite Element*. Ed. by Svenja Schoeder. 9th ed. Lehrstuhl für Numerische Mechanik, Fakultät für Maschinenwesen, Technische Universität München.

Fred L. Drake, Jr. (2016). *The Python Language Reference*. 3.51. Python Software Foundation.

Mckinney, Wes (2016). *Pandas: powerful Python data analysis toolkit*. Release 0.18.1. PyData Development Team.

Numpy (2016). *Numpy User Guide*. Release 1.11.0. Numpy community.

Rutzmoser, Johannes (2016). “MEMO: Nichtlineare Finite Elemente”.

Disclaimer

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

Garching, 30. 05. 2016 _____
(Signature)