



Yifei Wang

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

Semesterarbeit

30. 05. 2016

Supervisor:

Johannes Rutzmoser

Abstract

Bla bla bla

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

1	Introduction	1
1.1	Motivation	1
1.2	Structure of Study	2
2	Numerical Aspect for Finite Element Formulation	5
2.1	Shape Function	5
2.1.1	Quadrilateral Elements	5
2.1.2	Triangular and Tetrahedral Elements	6
2.2	Non-linear Element Formulation	6
2.2.1	Discretization of the Displacement Field	7
2.3	Non-linear Formulation	8
2.3.1	Gauss Integration	13
2.4	Extrapolation from Gauss Points	13
2.4.1	Quadrangle with quadratic shape function and eight nodes: Quad4 . . .	13
2.4.2	Quadrangle with quadratic shape function and eight nodes: Quad8 . . .	18
2.4.3	Triangle with three nodes: Tri3	20
2.4.4	Second order triangle with six nodes: Tri6	22
2.4.5	Tetrahedron element with four nodes: Tet4	25
2.4.6	Tetrahedron element with ten nodes: Tet10	26
2.5	Assembly	28
3	Programming Implementation	35
3.1	Pre-Processing	35
4	Element Type Test	39
4.1	Unit Testing with Python	39
4.2	The Patch Test	40
4.3	Complex Model Test	40
4.4	Convergence Analysis	91
	References	93

List of Figures

1.1	Stress-strain curve for steel	2
1.2	Flow chart of structure	3
1.3	Overview of AMfe structure	4
2.1	Linear shape function and quadratic shape function.	6
2.2	triangular and tetradral elemnets.	7
2.3	Quad4 in element coordinate and Gauss element coordinate.	15
2.4	Equation of side opposite corner 1 for Quad4.	15
2.5	Equation of side opposite corner 1 for Quad4.	16
2.6	Quad8 in element coordinate and Gauss element coordinate.	20
2.7	Equation of side opposite corner 1 for Quad8.	21
2.8	Tri3 in element coordinate and Gauss element coordinate.	22
2.9	Equation of side opposite corner 1 for Tri3.	24
2.10	Tri6 in element coordinate and Gauss element coordinate.	24
2.11	Equation of side opposite corner 1 for Tri6.	25
2.12	Tet4 in element coordinate and Gauss element coordinate.	26
2.13	Tet10 in element coordinate and Gauss element coordinate.	28
2.14	Two Tri6 elements in global and local perspective.	32
3.1	A simple meshed geometry	36
4.1	The displacement patch test with Tri6 element	41
4.2	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	41
4.3	Mesh with Tri3. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	43
4.4	Mesh with Tri3. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	44
4.5	Mesh with Tri3. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	45
4.6	Mesh with Tri3. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	46
4.7	Mesh with Tri3. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	47

4.8	Mesh with Tri3. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	48
4.9	Mesh with Tri6. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	49
4.10	Mesh with Tri6. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	50
4.11	Mesh with Tri6. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	51
4.12	Mesh with Tri6. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	52
4.13	Mesh with Tri6. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	53
4.14	Mesh with Tri6. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	54
4.15	Mesh with Quad4. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	55
4.16	Mesh with Quad4. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	56
4.17	Mesh with Quad4. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	57
4.18	Mesh with Quad4. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	58
4.19	Mesh with Quad4. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	59
4.20	Mesh with Quad4. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	60
4.21	Mesh with Quad8. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	61
4.22	Mesh with Quad8. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	62
4.23	Mesh with Quad8. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	63
4.24	Mesh with Quad8. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	64
4.25	Mesh with Quad8. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	65
4.26	Mesh with Quad8. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	66
4.27	Mesh with Tet4. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	67
4.28	Mesh with Tet4. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	68

4.29	Mesh with Tet4. upper: contour plot of ϵ_{zz} from ANSYS; lower: contour plot of ϵ_{zz} calculated in AMfe, demonstrate in ParaView	69
4.30	Mesh with Tet4. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	70
4.31	Mesh with Tet4. upper: contour plot of ϵ_{yz} from ANSYS; lower: contour plot of ϵ_{yz} calculated in AMfe, demonstrate in ParaView	71
4.32	Mesh with Tet4. upper: contour plot of ϵ_{xz} from ANSYS; lower: contour plot of ϵ_{xz} calculated in AMfe, demonstrate in ParaView	72
4.33	Mesh with Tet4. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	73
4.34	Mesh with Tet4. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	74
4.35	Mesh with Tet4. upper: contour plot of σ_{zz} from ANSYS; lower: contour plot of σ_{zz} calculated in AMfe, demonstrate in ParaView	75
4.36	Mesh with Tet4. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	76
4.37	Mesh with Tet4. upper: contour plot of σ_{yz} from ANSYS; lower: contour plot of σ_{yz} calculated in AMfe, demonstrate in ParaView	77
4.38	Mesh with Tet4. upper: contour plot of σ_{xz} from ANSYS; lower: contour plot of σ_{xz} calculated in AMfe, demonstrate in ParaView	78
4.39	Mesh with Tet10. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView	79
4.40	Mesh with Tet10. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView	80
4.41	Mesh with Tet10. upper: contour plot of ϵ_{zz} from ANSYS; lower: contour plot of ϵ_{zz} calculated in AMfe, demonstrate in ParaView	81
4.42	Mesh with Tet10. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView	82
4.43	Mesh with Tet10. upper: contour plot of ϵ_{yz} from ANSYS; lower: contour plot of ϵ_{yz} calculated in AMfe, demonstrate in ParaView	83
4.44	Mesh with Tet10. upper: contour plot of ϵ_{xz} from ANSYS; lower: contour plot of ϵ_{xz} calculated in AMfe, demonstrate in ParaView	84
4.45	Mesh with Tet10. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView	85
4.46	Mesh with Tet10. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView	86
4.47	Mesh with Tet10. upper: contour plot of σ_{zz} from ANSYS; lower: contour plot of σ_{zz} calculated in AMfe, demonstrate in ParaView	87
4.48	Mesh with Tet10. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView	88
4.49	Mesh with Tet10. upper: contour plot of σ_{yz} from ANSYS; lower: contour plot of σ_{yz} calculated in AMfe, demonstrate in ParaView	89

LIST OF FIGURES

ix

4.50 Mesh with Tet10. upper: contour plot of σ_{xz} from ANSYS; lower: contour plot of σ_{xz} calculated in AMfe, demonstrate in ParaView	90
4.51 Convergence analysis	91

List of Tables

2.1	Gauss-Legendre points and weights	14
2.2	Natural Coordinate of Quad4	17
2.3	Natural Coordinate of Tri3	23
2.4	Tetrahedral Coordinate of Tet4	27
2.5	Coordinate system of Tet10	29
3.1	Node list exported from ANSYS	37
3.2	Element list exported from ANSYS	37

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 analyse FEM

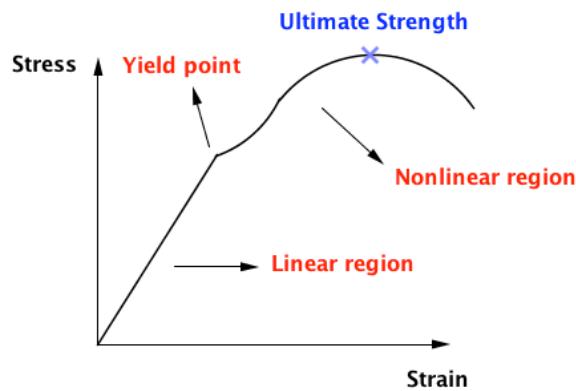


Figure 1.1: Stress-strain curve for steel

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 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.

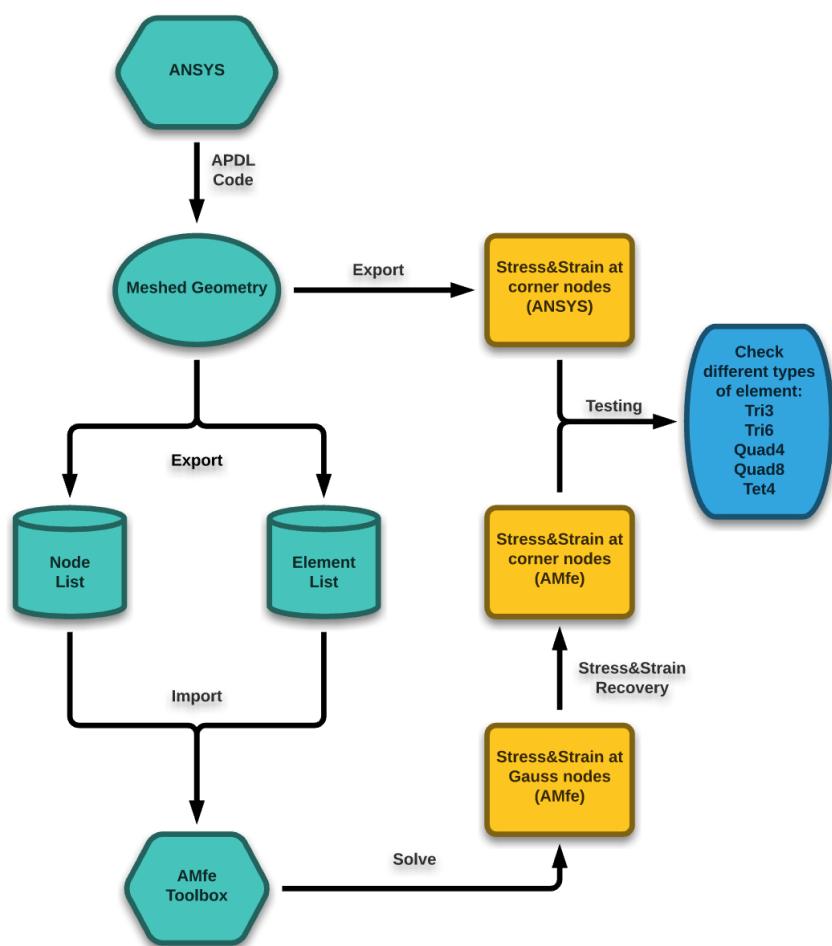


Figure 1.2: Flow chart of structure

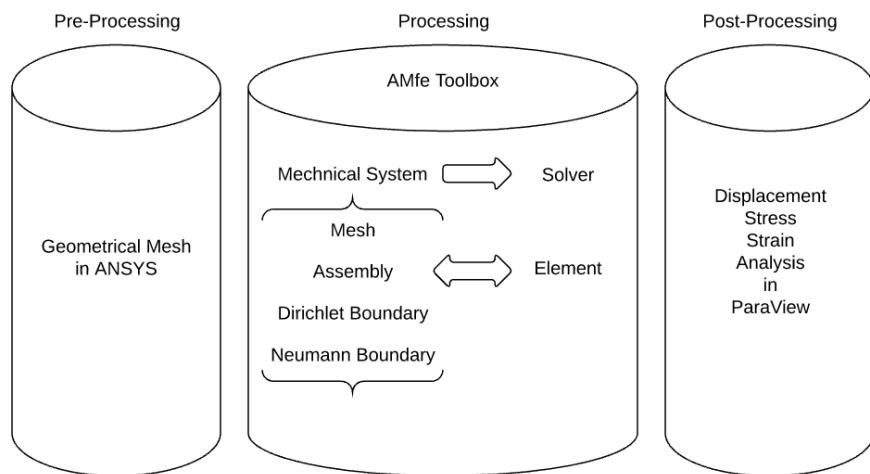


Figure 1.3: Overview of AMfe structure

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. 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. In one element, there are n nodes and n shape functions, in which 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:

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

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

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

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

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

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

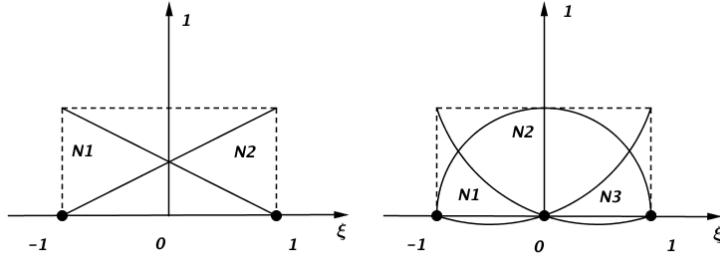


Figure 2.1: Linear shape function and quadratic shape function.

Lagrange polynomials are a general group of polynomials with the node undermentioned association property: a Lagrange polynomial $l_k^n - 1$ (in one coordinate ξ) of order $n - 1$ passes through n nodes with coordinates ξ^j ($j = 1, \dots, n$) of which the single node k equals one ($l_k^{n-1}(\xi^k) = 1$) and every other node results in zero ($l_k^{n-1}(\xi^j) = 0$ for all $j \neq k$).

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

$$k = 1, 2, \dots, n$$

For instance, to calculate linear, i.e. order 1, Lagrange polynomials can be calculated by $n - 1 = 1 \Rightarrow n = 2$. Using $\xi^1 = -1$ and $\xi^2 = +1$ results in: $l_1^1(\xi) = -1/2(\xi - 1)$ 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

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. These coordinates represent area and volume fractions, and visualization is given below:

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

$$\sum L_i = 1 \quad \sum V_i = 1$$

2.2 Non-linear Element Formulation

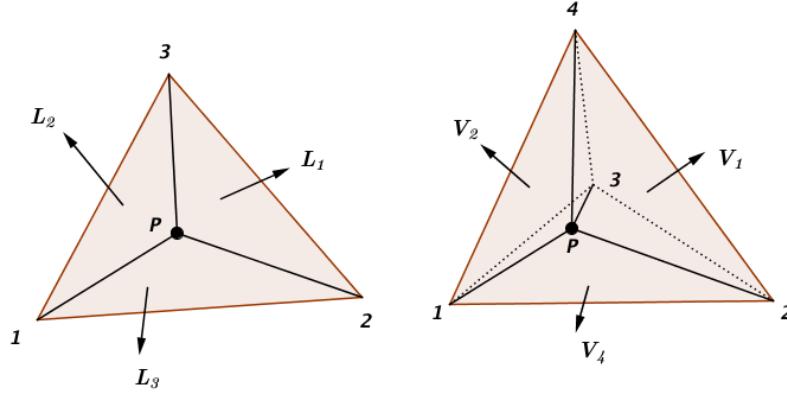


Figure 2.2: triangular and tetradral elemnets.

2.2.1 Discretization of the Displacement Field

After meshing the geometry, We discretize displacement field u by inserting shape function N . Firstly, we introduce the nodal displacement d^e to record displacements at each node of the considered element in each spatial direction. In elastic materials, It can be derived through the elastic constitutive equations that stress σ are directly related to strain ε at each point:

$$\sigma = E\varepsilon \quad (2.1)$$

It follows that the stress calculation procedure begins with strain calculation, and that the accuracy of stresses depends on that of strains. In the following sections, we focus our attention on two-dimensional isoparametric elements, as the calculation of strains, stresses and axial forces in bar elements is straightforward. Suppose that we have solved the master stiffness equations.

$$Ku = f \quad (2.2)$$

for the node displacements u . To calculate strains and stresses we perform a loop over all defined elements. Let ε be the element index of a specific two-dimensional isoparametric element encountered during this loop, and $u(\varepsilon)$ the vector of computed element node displacements. The strains at any point in the element can be related to these displacements as

$$\varepsilon = Bu(\varepsilon) \quad (2.3)$$

where B is the strain-displacement matrix assembled with the x and y derivatives of the element shape functions evaluated at the point where we are calculating strains. The corresponding stresses are given by

$$\sigma = E\varepsilon = EBu \quad (2.4)$$

In the applications it is of interest to evaluate and report these stresses at the element nodal points located on the corners and possibly midpoints of the element. These are called element nodal

point stresses. It is important to realize that the stresses computed at the same nodal point from adjacent elements will not generally be the same, since stresses are not required to be continuous in displacement-assumed finite elements. This suggests some form of stress averaging can be used to improve the stress accuracy, and indeed this is part of the stress recovery technique further. The results from this averaging procedure are called nodal point stresses.

2.3 Non-linear 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 the work of Johann Rutzmoser[Rutzmoser 2016] in non-linear 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:

$$\mathbf{N}(\xi) = \begin{pmatrix} N_1(\xi) \\ N_2(\xi) \\ N_3(\xi) \\ N_4(\xi) \\ N_5(\xi) \end{pmatrix} \quad (2.5)$$

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

$$\mathbf{X}^e = \begin{pmatrix} X_1 & Y_1 & Z_1 \\ X_2 & Y_2 & Z_2 \\ X_3 & Y_3 & Z_3 \\ X_4 & Y_4 & Z_5 \\ X_5 & Y_5 & Z_6 \end{pmatrix} \quad (2.6)$$

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

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = (X^e)^T N = \begin{pmatrix} X_1 & X_2 & X_3 & X_4 & X_5 \\ Y_1 & Y_2 & Y_3 & Y_4 & Y_5 \\ Z_1 & Z_2 & Z_3 & Z_4 & Z_5 \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{pmatrix} \quad (2.7)$$

The displacements \mathbf{u} at any points of the element are interpolated from the nodal coordinate, in the same manner that was completed for the coordinate X above. Then the displacement field can be shown as:

$$\begin{pmatrix} u_x(\xi) \\ u_y(\xi) \\ u_z(\xi) \end{pmatrix} = (\mathbf{u}^e)^T N = \begin{pmatrix} u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} \\ u_{z1} & u_{z2} & u_{z3} & u_{z4} & u_{z5} \end{pmatrix} \begin{pmatrix} N_1 \\ N_2 \\ N_3 \\ N_4 \\ N_5 \end{pmatrix} \quad (2.8)$$

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:

$$\frac{\partial \mathbf{u}}{\partial X} = (\mathbf{u}^e)^T \frac{\partial N}{\partial X} = (\mathbf{u}^e)^T \tilde{B}_0 \quad (2.9)$$

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

$$\frac{\partial \mathbf{N}}{\partial X} = \tilde{\mathbf{B}}_0 = \begin{pmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial Z} \\ \frac{\partial N_2}{\partial X} & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial Z} \\ \frac{\partial N_3}{\partial X} & \frac{\partial N_3}{\partial Y} & \frac{\partial N_3}{\partial Z} \\ \frac{\partial N_4}{\partial X} & \frac{\partial N_4}{\partial Y} & \frac{\partial N_4}{\partial Z} \\ \frac{\partial N_5}{\partial X} & \frac{\partial N_5}{\partial Y} & \frac{\partial N_5}{\partial Z} \end{pmatrix} = \frac{\partial \mathbf{N}}{\partial \xi} \frac{\partial \xi}{\partial X} \quad (2.10)$$

The first term $\frac{\partial \mathbf{N}}{\partial \xi}$ in Equation 2.10 can be directly expressed with shape function relating coordinate ξ . It is shown as:

$$\frac{\partial \mathbf{N}}{\partial \xi} = \begin{pmatrix} \frac{\partial N_1}{\partial \xi_1} & \frac{\partial N_1}{\partial \xi_2} & \frac{\partial N_1}{\partial \xi_3} \\ \frac{\partial N_2}{\partial \xi_1} & \frac{\partial N_2}{\partial \xi_2} & \frac{\partial N_2}{\partial \xi_3} \\ \frac{\partial N_3}{\partial \xi_1} & \frac{\partial N_3}{\partial \xi_2} & \frac{\partial N_3}{\partial \xi_3} \\ \frac{\partial N_4}{\partial \xi_1} & \frac{\partial N_4}{\partial \xi_2} & \frac{\partial N_4}{\partial \xi_3} \\ \frac{\partial N_5}{\partial \xi_1} & \frac{\partial N_5}{\partial \xi_2} & \frac{\partial N_5}{\partial \xi_3} \end{pmatrix} \quad (2.11)$$

The second term $\frac{\partial \xi}{\partial X}$ is not straightforward to obtain. However, it is much easier to get the matrix $\frac{\partial X}{\partial \xi}$. This matrix is called Jacobian matrix \mathbf{J} , which is a matrix of ξ with respect to x . The inverse Jacobian matrix is denoted as \mathbf{J}^{-1} , which is the second term in the Equation 2.10.

The deformation gradient \mathbf{F} describes the mapping of an infinitesimal fiber from its initial state to its new position in the current configuration. $\mathbf{F} = \mathbf{H} + \mathbf{I}$ can be derived through the transient matrix $\mathbf{H} = \frac{\partial \mathbf{u}}{\partial X} = (\mathbf{u}^e)^T \tilde{\mathbf{B}}_0$. The Green-Lagrange strain tensor can also be written as:

$$\mathbf{E} = \frac{1}{2} (\mathbf{H} + \mathbf{H}^T + \mathbf{H}^T \mathbf{H}) \quad (2.12)$$

$$\mathbf{E} = \frac{1}{2} (\mathbf{F}^T \mathbf{F} - \mathbf{I}) \quad (2.13)$$

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

$$\mathbf{H} = \frac{\partial \mathbf{u}}{\partial X} = \frac{\partial \mathbf{u}}{\partial \xi} \frac{\partial \xi}{\partial X} = (\mathbf{u}^e)^T \frac{\partial \mathbf{N}}{\partial \xi} \left[(\mathbf{X}^e)^T \frac{\partial \mathbf{N}}{\partial \xi} \right]^{-1} \quad (2.14)$$

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

$$\mathbf{S} = \mathbf{C} : \mathbf{E} \quad (2.15)$$

The value of stress is subsequently transferred into the degree of freedom 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 $\delta W_{int} = \int \boldsymbol{\sigma} : \delta \mathbf{e} d\Omega_0 = \int \mathbf{S} : \delta \mathbf{E} d\Omega_0$ equals the external energy $\delta W_{ext} = (\delta \mathbf{u}^{e,v})^T \mathbf{f}_{nl}^v$, which is caused by external nodal force. The complete equation can be expressed as:

$$\delta W = \delta W_{ext} = (\delta \mathbf{u}^{e,v})^T \mathbf{f}_{int}^v = \int \mathbf{S} : \delta \mathbf{E} d\Omega_0 = \int (\delta \mathbf{E}^v)^T \mathbf{S}^v d\Omega_0 \quad (2.16)$$

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 $\delta \mathbf{E}$. From equation 2.13, it is obvious that the variation of tensor $\delta \mathbf{E}$ is determined by the variation of deformation gradient. According to above equation 2.14, the variation of deformation gradient can be transformed into:

$$\delta \mathbf{F} = \delta \mathbf{H} = (\delta \mathbf{u}^e)^T \frac{\delta \mathbf{N}}{\delta \mathbf{X}} = (\delta \mathbf{u}^e)^T \tilde{\mathbf{B}}_0 \quad (2.17)$$

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

$$\delta \mathbf{E} = \frac{1}{2} (\delta \mathbf{F}^T \mathbf{F} + \mathbf{F}^T \delta \mathbf{F}) \quad (2.18)$$

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

$$\delta \mathbf{E}^v = \mathbf{B}_0 \delta \mathbf{u}^{e,v} \quad (2.19)$$

The entry of \mathbf{B}_0 will be packed into a black box and directly implemented in the FEM simulation. By solving simultaneous equations of Equation 2.16 and Equation 2.19, the non-linear force \mathbf{f}_{int}^v results in:

$$\delta W = (\delta \mathbf{u}^{e,v})^T \mathbf{f}_{int}^v = \int (\delta \mathbf{E}^v)^T \mathbf{S}^v d\Omega_0 = (\delta \mathbf{u}^{e,v})^T \int \mathbf{B}_0^T \mathbf{S}^v d\Omega_0 \quad (2.20)$$

$$\mathbf{f}_{int}^v = \int \mathbf{B}_0^T \mathbf{S}^v d\Omega_0 \quad (2.21)$$

The non-linear internal force of element concerning the coordinate of node can be integrated by \mathbf{B}_0 -matrix 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 f_{int}^v in respect to the nodal degree of freedom $\mathbf{u}^{e,v}$:

$$\frac{\partial f_{int}^v}{\partial \mathbf{u}^{e,v}} = \mathbf{K} = \int \frac{\partial \mathbf{B}_0^T}{\partial \mathbf{u}^{e,v}} \mathbf{S}^v d\Omega_0 + \int \mathbf{B}_0^T \frac{\partial \mathbf{S}^v}{\partial \mathbf{u}^{e,v}} d\Omega_0 = \mathbf{K}_{geo} + \mathbf{K}_{mat} \quad (2.22)$$

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

$$\mathbf{K}_{mat} = \int \mathbf{B}_0^T \frac{\partial \mathbf{S}^v}{\partial \mathbf{u}^{e,v}} d\Omega_0 = \int \mathbf{B}_0^T \frac{\partial \mathbf{S}^v}{\partial \mathbf{E}^v} \frac{\partial \mathbf{E}^v}{\partial \mathbf{u}^{e,v}} d\Omega_0 = \int \mathbf{B}_0^T \mathbf{C}^{SE} \mathbf{B}_0 d\Omega_0 \quad (2.23)$$

The derivative of stress in respect to strain is considered as Tangent-Modulus $\mathbf{C}^{SE} = \frac{\partial \mathbf{S}^v}{\partial \mathbf{E}^v}$ by constitutive equation. The derivative of strain is \mathbf{B}_0 as determined in equation 2.13.

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 $\mathbf{P} = \mathbf{SF}^T$, the internal energy from Equation 2.16 is reformulated into:

$$\delta W = \mathbf{u}^e : f_{int} = \int \delta \mathbf{F}^T : \mathbf{P} d\Omega_0 = \int \delta \mathbf{F}_{ij} \mathbf{P}_{ji} d\Omega_0 \quad (2.24)$$

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

$$\delta W = \delta \mathbf{u}^e : f_{int} = \mathbf{u}_{ki}^e f_{ki}^{int} = \int \delta \mathbf{F}_{ji} \mathbf{P}_{ij} d\Omega_0 = \delta \mathbf{u}_{ki}^e \int \tilde{\mathbf{B}}_{kj}^0 \mathbf{P}_{ji} d\Omega_0 = \delta \mathbf{u}^e : \int \tilde{\mathbf{B}}_0 \mathbf{P} d\Omega_0 \quad (2.25)$$

$$f_{int} = \int \tilde{\mathbf{B}}_0 \mathbf{P} d\Omega_0 \quad (2.26)$$

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

$$\dot{f}_{int} = \int \tilde{\mathbf{B}}_0 \dot{\mathbf{S}} \mathbf{F}^T d\Omega_0 + \int \tilde{\mathbf{B}}_0 \mathbf{S} \dot{\mathbf{F}}^T d\Omega_0 \quad (2.27)$$

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 corresponds to geometrical stiffness, as is shown in equation 2.22. Because the time derivative $\dot{\mathbf{F}}$ equals $\dot{\mathbf{u}}^{eT} \tilde{\mathbf{B}}_0$, Equation 2.27 can be transformed as:

$$\dot{f}_{int,geo} = \int \tilde{\mathbf{B}}_0 \mathbf{S} \tilde{\mathbf{B}}_0 \Omega_0 \dot{\mathbf{u}}^e \quad (2.28)$$

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

$$\dot{\mathbf{f}}_{int,geo} = \mathbf{K}_{geo}\dot{\mathbf{u}}^e = \int \tilde{\mathbf{B}}_0 S \tilde{\mathbf{B}}_0 \Omega_0 \dot{\mathbf{u}}^e \quad (2.29)$$

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:

$$\int f(x) d\Omega = \int f(x) \frac{\partial \xi}{\partial x} d\Omega = \int f(x) \frac{\partial \Omega}{\partial \xi} d\xi = \int f(x) \det \left(\frac{\partial X}{\partial \xi} \right) d\xi \quad (2.30)$$

2.3.1 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 $f(\xi)$ on the spatial domain Ω by replacing a summation of certain function values at the so-called Gauss points $\tilde{\xi}$ that are each multiplied by a scalar (weight) ω . A clear expression of Gauss integration is:

$$\int_E f(\xi) d\xi_1 d\xi_2 d\xi_3 = \sum_{i=1}^{numgp_1} \sum_{j=1}^{numgp_2} \sum_{k=1}^{numgp_3} f(\tilde{\xi}_1^i \tilde{\xi}_2^j \tilde{\xi}_3^k) \cdot w_1^i w_2^j w_3^k \quad (2.31)$$

In this expression, we used corresponding weights for each of the three spatial directions, which allows us to use Gauss points coordinates and weights tabulated for the one-dimensional case [Wall 2014]. The locations and weights of these points can be expressed analytically in Table 2.1.

2.4 Extrapolation from Gauss Points

2.4.1 Quadrangle with quadratic shape function and eight 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 having two degrees of freedom at each node: translations in the nodal x and y directions. The element has plasticity, creep, swelling, large deflection, and large strain capabilities. The normal Gauss integration rule for element stiffness evaluation is 2×2 , as is illustrated in Figure 2.3. The components(strains, stresses) 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 2.3. 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^G . 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 2.3. The external element is denoted by (E). The internal Gauss

Table 2.1: Gauss-Legendre points and weights

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

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.

$$k_i^G = \frac{k_i^E}{\sqrt{3}}, \quad k_i^E = k_i^G \sqrt{3} \quad (2.32)$$

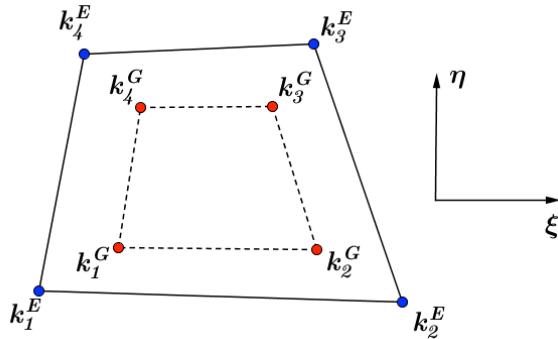


Figure 2.3: Quad4 in element coordinate and Gauss element coordinate.

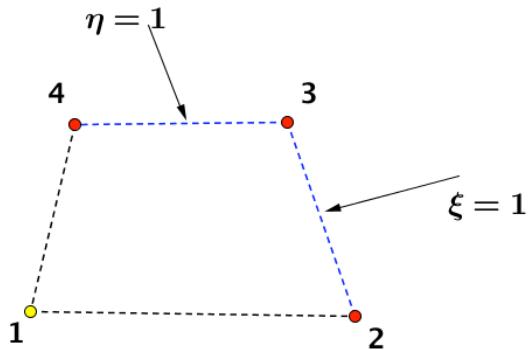


Figure 2.4: Equation of side opposite corner 1 for Quad4.

The element geometry and natural coordinates are shown in Figure 2.5. There is only one type of node and associated shape function. We consider node 1 to be a typical node. From Figure 2.4 we can suggest:

$$N_1^e = c_1 L_{2-3} L_{3-4} \quad (2.33)$$

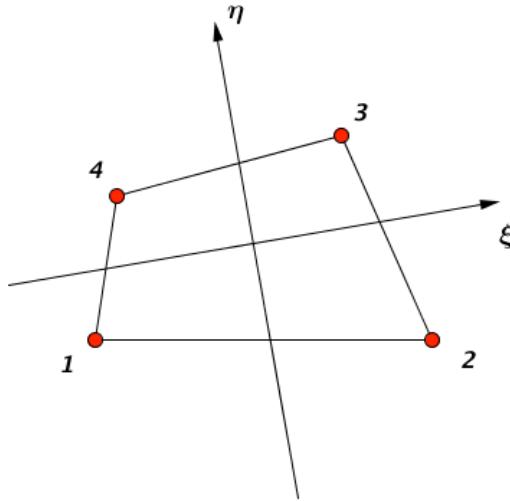


Figure 2.5: Equation of side opposite corner 1 for Quad4.

The equation of edge 2-3 is $\xi - 1 = 0$. The equation of edge 3-4 is $\eta - 1 = 0$. Inserting in Equation (2.33) results in:

$$N_1^e(\xi, \eta) = c_1 (\xi - 1)(\eta - 1) = c_1 (1 - \xi)(1 - \eta) \quad (2.34)$$

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

$$N_1^e(-1, -1) = c_1 \times 2 \times 2 = 4c_1 = 1 \quad (2.35)$$

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

$$N_1^e = \frac{1}{4} (1 - \xi)(1 - \eta) \quad (2.36)$$

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:

$$N_2^e = \frac{1}{4} (1 + \xi)(1 - \eta) \quad (2.37)$$

$$N_3^e = \frac{1}{4} (1 + \xi)(1 + \eta) \quad (2.38)$$

$$N_4^e = \frac{1}{4} (1 - \xi)(1 + \eta) \quad (2.39)$$

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

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

and Figure 2.3, we have the corner nodes in Gauss coordinate: $k_1^E(-\sqrt{3}, -\sqrt{3})$, $k_2^E(\sqrt{3}, -\sqrt{3})$, $k_3^E(\sqrt{3}, \sqrt{3})$, $k_4^E(\sqrt{3}, \sqrt{3})$. The extrapolation process is to replace all the corner nodes in Gauss coordinate with the shape function of each Gauss points. Then we obtain:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} \\ 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} \quad (2.40)$$

Here w' means strains and stresses for our case. So the w' part can be written as:

$$\begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} = \begin{pmatrix} \epsilon_{1xx} & \epsilon_{1yy} & \epsilon_{1zz} & 2\epsilon_{1yz} & 2\epsilon_{1xz} & 2\epsilon_{1xy} \\ \epsilon_{2xx} & \epsilon_{2yy} & \epsilon_{2zz} & 2\epsilon_{2yz} & 2\epsilon_{2xz} & 2\epsilon_{2xy} \\ \epsilon_{3xx} & \epsilon_{3yy} & \epsilon_{3zz} & 2\epsilon_{3yz} & 2\epsilon_{3xz} & 2\epsilon_{3xy} \\ \epsilon_{4xx} & \epsilon_{4yy} & \epsilon_{4zz} & 2\epsilon_{4yz} & 2\epsilon_{4xz} & 2\epsilon_{4xy} \end{pmatrix} \quad (2.41)$$

$$\begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} = \begin{pmatrix} \sigma_{1xx} & \sigma_{1yy} & \sigma_{1zz} & \sigma_{1yz} & \sigma_{1xz} & \sigma_{1xy} \\ \sigma_{2xx} & \sigma_{2yy} & \sigma_{2zz} & \sigma_{2yz} & \sigma_{2xz} & \sigma_{2xy} \\ \sigma_{3xx} & \sigma_{3yy} & \sigma_{3zz} & \sigma_{3yz} & \sigma_{3xz} & \sigma_{3xy} \\ \sigma_{4xx} & \sigma_{4yy} & \sigma_{4zz} & \sigma_{4yz} & \sigma_{4xz} & \sigma_{4xy} \end{pmatrix} \quad (2.42)$$

2.4.2 Quadrangle with quadratic shape function and eight nodes: Quad8

Quad8 is a higher order 2D, 8 node element. This element is defined by 8 nodes having two degrees of freedom 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. 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 2.6

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 2.4.1. Here we summarize the calculation for three typical types, represented by nodes 1, 5 and 9. The three types are illustrated in Figure 2.7.

$$N_1^e = c_1 L_{2-3} L_{3-4} L_{5-7} L_{6-8} = c_1 (\xi - 1)(\eta - 1) \xi \eta \quad (2.43)$$

$$N_5^e = c_5 L_{2-3} L_{1-4} L_{6-8} L_{3-4} = c_5 (\xi - 1)(\xi + 1) \eta (\eta - 1) = c_5 (1 - \xi^2) \eta (1 - \eta) \quad (2.44)$$

$$N_9^e = c_9 L_{1-2} L_{2-3} L_{3-4} L_{4-1} = c_9 (\xi - 1)(\eta - 1)(\xi + 1)(\eta + 1) = c_9 (1 - \xi^2)(1 - \eta^2) \quad (2.45)$$

By applying the normalization conditions, the results are:

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

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

$$N_1 = \frac{1}{4} (\xi - 1)(\eta - 1) \cdot \xi \cdot \eta \quad (2.46)$$

$$N_2 = \frac{1}{4} (\xi + 1)(\eta - 1) \cdot \xi \cdot \eta \quad (2.47)$$

$$N_3 = \frac{1}{4} (\xi + 1)(\eta + 1) \cdot \xi \cdot \eta \quad (2.48)$$

$$N_4 = \frac{1}{4} (\xi - 1)(\eta + 1) \cdot \xi \cdot \eta \quad (2.49)$$

$$N_5 = \frac{1}{2} (1 + \xi)(1 + \xi)(\eta - 1) \cdot \eta \quad (2.50)$$

$$N_6 = \frac{1}{2} (1 + \eta)(1 - \eta)(\xi + 1) \cdot \xi \quad (2.51)$$

$$N_7 = \frac{1}{2} (1 + \xi)(1 + \xi)(\eta + 1) \cdot \eta \quad (2.52)$$

$$N_8 = \frac{1}{2} (1 + \eta)(1 - \eta)(\xi - 1) \cdot \xi \quad (2.53)$$

$$N_9 = (1 + \xi)(1 - \xi)(1 + \eta)(1 - \eta) \quad (2.54)$$

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

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_3 & a_2 & a_4 & a_5 & a_5 & a_4 & a_6 \\ a_2 & a_1 & a_2 & a_3 & a_4 & a_4 & a_5 & a_5 & a_6 \\ a_3 & a_2 & a_1 & a_2 & a_5 & a_4 & a_4 & a_5 & a_6 \\ a_2 & a_3 & a_2 & a_1 & a_5 & a_5 & a_4 & a_4 & a_6 \\ 0 & 0 & 0 & 0 & a_7 & 0 & a_8 & 0 & a_9 \\ 0 & 0 & 0 & 0 & 0 & a_7 & 0 & a_8 & a_9 \\ 0 & 0 & 0 & 0 & a_8 & 0 & a_7 & 0 & a_9 \\ 0 & 0 & 0 & 0 & 0 & a_8 & 0 & a_7 & a_9 \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \\ w_5' \\ w_6' \\ w_7' \\ w_8' \\ w_9' \end{pmatrix} \quad (2.55)$$

$$a_1 = +2.1869398$$

$$a_2 = +0.2777778$$

$$a_3 = +0.0352824$$

$$a_4 = -0.9858870$$

$$a_5 = -0.1252241$$

$$a_6 = +0.4444444$$

$$a_7 = +1.4788331$$

$$a_8 = +0.1878361$$

$$a_9 = -0.6666666$$

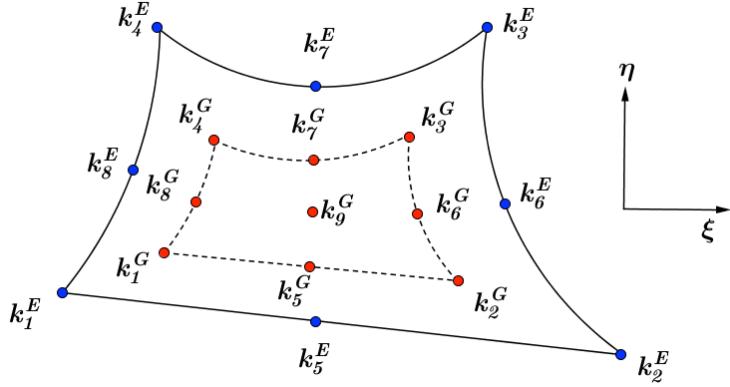


Figure 2.6: Quad8 in element coordinate and Gauss element coordinate.

2.4.3 Triangle with three nodes: Tri3

Tri3 is used for 2D modelling of solid structure. This element is defined by 3 nodes having two degrees of freedom at each node: translation in the nodal x and y direction. The geometry of a 3-node triangle, as is shown in Figure 2.8, 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. 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 method: the equation of the triangle edge corresponding to node i is $L_{j-k} = \xi_i = 0$, in which j and k are the cyclic permutations of i . See Figure 2.9 for $i = 1, j = 2$ and $k = 3$. Hence the obvious general formula is:

$$N_i^e = c_i L_i \quad (2.56)$$

To find c_1 , evaluate $N_1^e(\xi_1, \xi_2, \xi_3)$ at node 1. The triangular coordinates of this node are $\xi_1 = 1$, $\xi_2 = \xi_3 = 0$. $N_1^e(1, 0, 0) = c_1 \times 1 = 1$. Therefore $c_1 = 1$, using the same method, $c_2 = 1$, $c_3 = 1$. Thus the shape functions are:

$$N_1^e = L_1, \quad N_2^e = L_2, \quad N_3^e = L_3 \quad (2.57)$$

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

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} 5/3 & -1/3 & -1/3 \\ -1/3 & 5/3 & -1/3 \\ -1/3 & -1/3 & 5/3 \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \end{pmatrix} \quad (2.58)$$

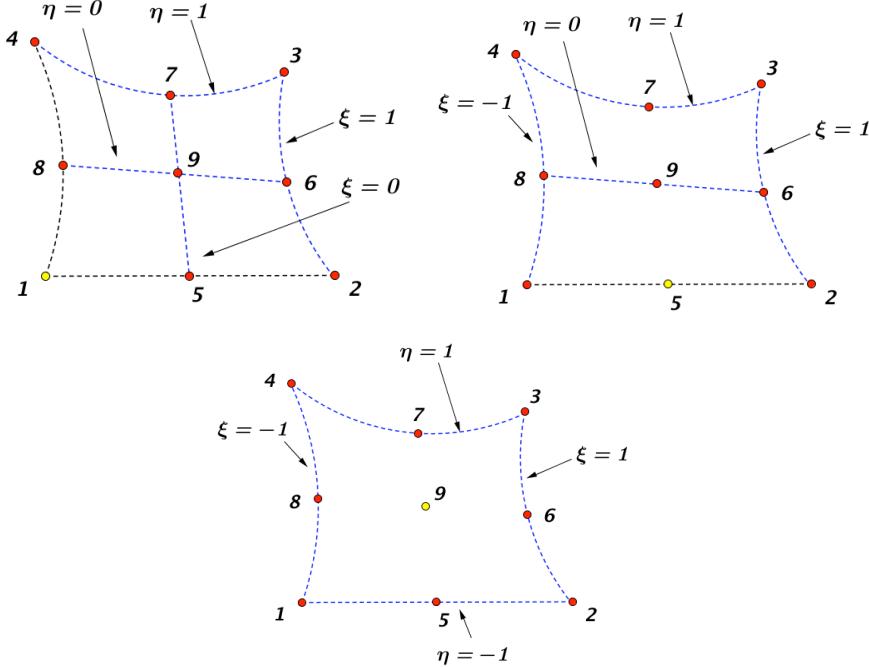


Figure 2.7: Equation of side opposite corner 1 for Quad8.

Quantities that are closely related to the element geometry are best represented in triangular coordinates. However, quantities such as displacements, strains and stresses are often calculated in the Cartesian system. Thus, we need transformation equations through to make it possible to pass from one coordinate system to the other. Cartesian and triangular coordinates are linked by the following relation:

$$\begin{pmatrix} 1 \\ x \\ y \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} \quad (2.59)$$

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} = \frac{1}{2A} \begin{pmatrix} 2A_{23} & y_{23} & x_{32} \\ 2A_{31} & y_{31} & x_{13} \\ 2A_{12} & y_{12} & x_{21} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix} \quad (2.60)$$

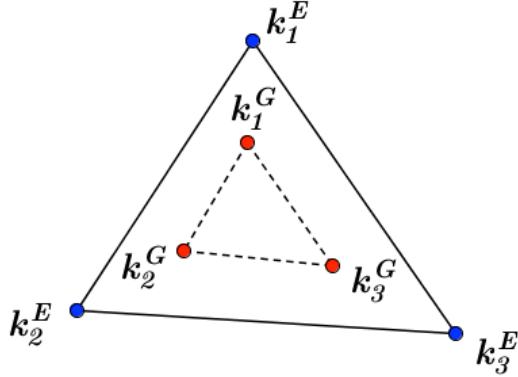


Figure 2.8: Tri3 in element coordinate and Gauss element coordinate.

2.4.4 Second order triangle with six nodes: Tri6

Tri6 is a higher order 2D, 6 node element. This element is defined by 6 nodes having two degrees of freedom at each node: translation in the nodal x and y direction. The geometry of Tri6 is shown in Figure 2.10. 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. Inspection reveals two types of nodes: corners(1, 2 and 3) and midside nodes(4, 5 and 6). In both cases, it is necessary to get the product of the two linear functions in the triangular coordinates because the shape function should be quadratic. These functions are illustrated in Figure 2.11 for corner node 1 and midside node 4, respectively.

For corner node 1, inspection of Figure 2.11 at left hand side suggests trying

$$N_1^e = c_1 L_{2-3} L_{4-6} \quad (2.61)$$

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 2.61 are:

$$L_{2-3} : \xi_1 = 0, \quad L_{4-6} : \xi_1 - \frac{1}{2} = 0 \quad (2.62)$$

Replacing the above functions into Equation 2.61

$$N_1^e = c_1 \xi_1 \left(\xi_1 - \frac{1}{2} \right) \quad (2.63)$$

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 finally:

$$N_1^e = 2 \xi_1 \left(\xi_1 - \frac{1}{2} \right) = \xi_1 (2\xi_1 - 1) \quad (2.64)$$

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

For midside node 4, observation of Figure 2.11 suggests trying:

$$N_4^e = c_4 L_{2-3} L_{1-3} \quad (2.65)$$

The equations of edges L_{2-3} and L_{1-3} are $\xi_1 = 0$ and $\xi_2 = 0$, respectively. Therefore, $N_4^e(\xi_1, \xi_2, \xi_3) = c_4 \xi_1 \xi_2$. 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 is:

$$N_4^e = 4\xi_1 \xi_2 \quad (2.66)$$

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

$$N_2^e = \xi_2 (2\xi_2 - 1) \quad (2.67)$$

$$N_3^e = \xi_3 (2\xi_3 - 1) \quad (2.68)$$

$$N_5^e = 4\xi_2 \xi_3 \quad (2.69)$$

$$N_6^e = 4\xi_1 \xi_3 \quad (2.70)$$

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

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{pmatrix} = \begin{pmatrix} a_1 & a_1 & a_2 & a_3 & a_4 & a_4 \\ a_1 & a_2 & a_1 & a_4 & a_4 & a_3 \\ a_2 & a_1 & a_1 & a_4 & a_3 & a_4 \\ a_1 & a_7 & a_7 & a_5 & a_6 & a_5 \\ a_7 & a_7 & a_1 & a_6 & a_5 & a_5 \\ a_7 & a_1 & a_7 & a_5 & a_5 & a_6 \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \\ w_5' \\ w_6' \end{pmatrix} \quad (2.71)$$

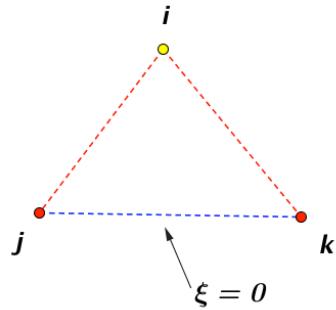


Figure 2.9: Equation of side opposite corner 1 for Tri3.

$$a_1 = +0.55555556$$

$$a_2 = +3.88888889$$

$$a_3 = +0.44444444$$

$$a_4 = -2.22222222$$

$$a_5 = -0.88888889$$

$$a_6 = +1.77777778$$

$$a_7 = +0.22222222$$

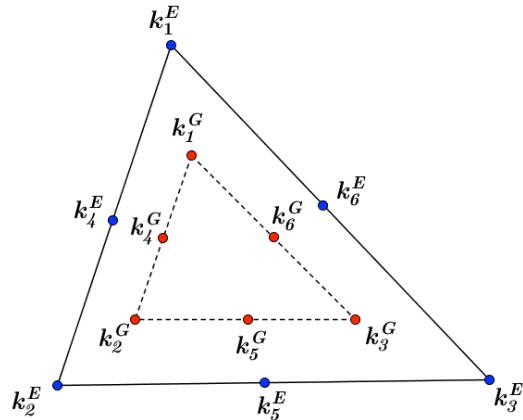


Figure 2.10: Tri6 in element coordinate and Gauss element coordinate.

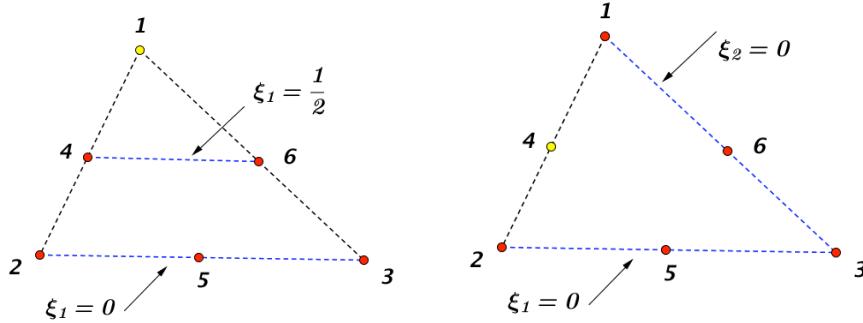


Figure 2.11: Equation of side opposite corner 1 for Tri6.

2.4.5 Tetrahedron element with four nodes: Tet4

Tet4 is a higher order 3D 4-node solid element. The element is defined by 4 nodes having three degrees of freedom per node: translation in the nodal x, y and z directions. The geometry of Tet4 is shown in Figure 2.12. For the tetrahedron element, it is beneficial to use its own coordinate. The shape functions of Tet4 are:

$$N_1 = L_1 \quad (2.72)$$

$$N_2 = L_2 \quad (2.73)$$

$$N_3 = L_3 \quad (2.74)$$

$$N_4 = L_4 \quad (2.75)$$

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: $k_1^E(j, k, k, k)$, $k_2^E(k, j, k, k)$, $k_3^E(k, k, j, k)$, $k_4^E(k, k, k, j)$ ($j = 1.927051$, $k = -0.309017$). After inserting the corner points into shape function, the extrapolation of Tet4 is shown as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_2 & a_2 \\ a_2 & a_1 & a_2 & a_2 \\ a_2 & a_2 & a_1 & a_2 \\ a_2 & a_2 & a_2 & a_1 \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} \quad (2.76)$$

$$a_1 = +1.927051$$

$$a_2 = -0.309017$$

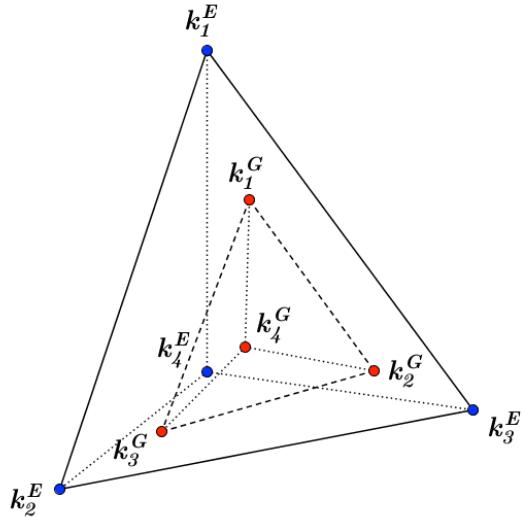


Figure 2.12: Tet4 in element coordinate and Gauss element coordinate.

2.4.6 Tetrahedron element with ten nodes: Tet10

Tet10 is a higher order 3D 10-node solid element. The element is defined by 10 nodes having three degrees of freedom 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 2.13. We choose a tetrahedron with four nodes as internal Gauss element. The shape function of Tet10 is same as Tet4:

$$N_1 = L_1 \quad (2.77)$$

$$N_2 = L_2 \quad (2.78)$$

$$N_3 = L_3 \quad (2.79)$$

$$N_4 = L_4 \quad (2.80)$$

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

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

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

be formed as:

$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \\ w_{10} \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_2 & a_2 \\ a_2 & a_1 & a_2 & a_2 \\ a_2 & a_2 & a_1 & a_2 \\ a_2 & a_2 & a_2 & a_1 \\ a_3 & a_3 & a_2 & a_2 \\ a_3 & a_2 & a_3 & a_2 \\ a_3 & a_2 & a_2 & a_3 \\ a_2 & a_3 & a_3 & a_2 \\ a_2 & a_3 & a_2 & a_3 \\ a_2 & a_2 & a_3 & a_3 \end{pmatrix} \begin{pmatrix} w_1' \\ w_2' \\ w_3' \\ w_4' \end{pmatrix} \quad (2.81)$$

$$a_1 = +1.927051$$

$$a_2 = -0.309017$$

$$a_3 = +0.809017$$

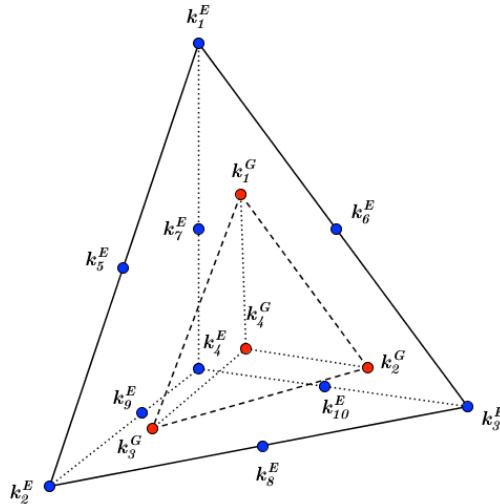


Figure 2.13: Tet10 in element coordinate and Gauss element coordinate.

2.5 Assembly

In the previous section, 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, element stiffness matrix and force vector are also assembled at each degree of freedom. 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 2.14. 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

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

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

given as:

$$k^{(1)} = \begin{pmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{33}^{(1)} & k_{34}^{(1)} & k_{35}^{(1)} & k_{36}^{(1)} \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{43}^{(1)} & k_{44}^{(1)} & k_{45}^{(1)} & k_{46}^{(1)} \\ k_{51}^{(1)} & k_{52}^{(1)} & k_{53}^{(1)} & k_{54}^{(1)} & k_{55}^{(1)} & k_{56}^{(1)} \\ k_{61}^{(1)} & k_{62}^{(1)} & k_{63}^{(1)} & k_{64}^{(1)} & k_{65}^{(1)} & k_{66}^{(1)} \end{pmatrix} \quad (2.82)$$

$$k^{(2)} = \begin{pmatrix} k_{11}^{(2)} & k_{12}^{(2)} & k_{13}^{(2)} & k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ k_{21}^{(2)} & k_{22}^{(2)} & k_{23}^{(2)} & k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\ k_{31}^{(2)} & k_{32}^{(2)} & k_{33}^{(2)} & k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\ k_{41}^{(2)} & k_{42}^{(2)} & k_{43}^{(2)} & k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\ k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\ k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)} \end{pmatrix} \quad (2.83)$$

Now we can assemble both stiffness matrix contribution, namely from Equation 2.82 and from Equation 2.83, at the same degree of freedom. The procedure of assembly is represented as follows:

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

$$= \begin{pmatrix} k_{11}^{(1)} & k_{12}^{(1)} & k_{15}^{(1)} & k_{16}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} & 0 & 0 \\ k_{21}^{(1)} & k_{22}^{(1)} & k_{25}^{(1)} & k_{26}^{(1)} & k_{23}^{(1)} & k_{24}^{(1)} & 0 & 0 \\ k_{51}^{(1)} & k_{52}^{(1)} & k_{55}^{(1)} + k_{11}^{(2)} & k_{56}^{(1)} + k_{12}^{(2)} & k_{53}^{(1)} + k_{13}^{(2)} & k_{54}^{(1)} + k_{14}^{(2)} & k_{15}^{(2)} & k_{16}^{(2)} \\ k_{61}^{(1)} & k_{62}^{(1)} & k_{65}^{(1)} + k_{21}^{(2)} & k_{66}^{(1)} + k_{22}^{(2)} & k_{63}^{(1)} + k_{23}^{(2)} & k_{64}^{(1)} + k_{24}^{(2)} & k_{25}^{(2)} & k_{26}^{(2)} \\ k_{31}^{(1)} & k_{32}^{(1)} & k_{35}^{(1)} + k_{31}^{(2)} & k_{36}^{(1)} + k_{32}^{(2)} & k_{33}^{(1)} + k_{33}^{(2)} & k_{34}^{(1)} + k_{34}^{(2)} & k_{35}^{(2)} & k_{36}^{(2)} \\ k_{41}^{(1)} & k_{42}^{(1)} & k_{45}^{(1)} + k_{41}^{(2)} & k_{46}^{(1)} + k_{42}^{(2)} & k_{43}^{(1)} + k_{43}^{(2)} & k_{44}^{(1)} + k_{44}^{(2)} & k_{45}^{(2)} & k_{46}^{(2)} \\ 0 & 0 & k_{51}^{(2)} & k_{52}^{(2)} & k_{53}^{(2)} & k_{54}^{(2)} & k_{55}^{(2)} & k_{56}^{(2)} \\ 0 & 0 & k_{61}^{(2)} & k_{62}^{(2)} & k_{63}^{(2)} & k_{64}^{(2)} & k_{65}^{(2)} & k_{66}^{(2)} \end{pmatrix} \quad (2.84)$$

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

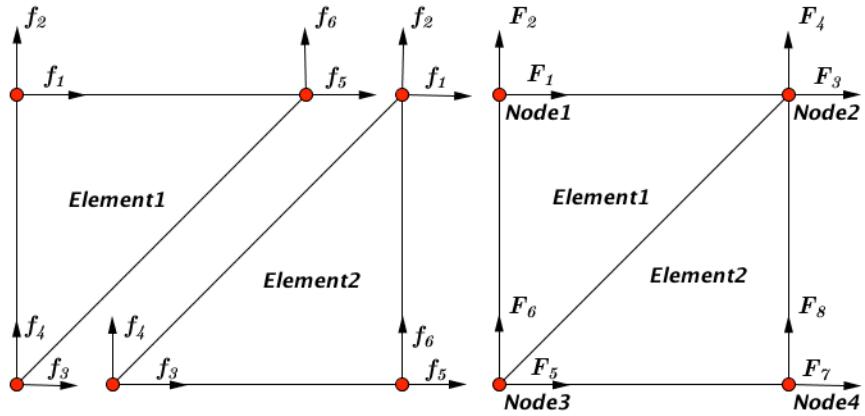


Figure 2.14: Two Tri6 elements in global and local perspective.

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

$$F = \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_5^{(1)} \\ f_6^{(1)} \\ f_3^{(1)} \\ f_4^{(1)} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ f_1^{(2)} \\ f_2^{(2)} \\ f_3^{(2)} \\ f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \end{pmatrix} = \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_5^{(1)} + f_1^{(2)} \\ f_6^{(1)} + f_2^{(2)} \\ f_3^{(1)} + f_3^{(2)} \\ f_4^{(1)} + f_4^{(2)} \\ f_5^{(2)} \\ f_6^{(2)} \end{pmatrix} \quad (2.85)$$

For strain and stress, we need to assemble the contribution 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:

$$\epsilon = \sum_{i=1}^n \frac{\epsilon_{i,raw}}{n} \quad \sigma = \sum_{i=1}^n \frac{\sigma_{i,raw}}{n} \quad (2.86)$$

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

$$\epsilon = \begin{pmatrix} \epsilon_{1xx}^{(1)} & \epsilon_{1yy}^{(1)} & \epsilon_{1zz}^{(1)} & 2\epsilon_{1yz}^{(1)} & 2\epsilon_{1xz}^{(1)} & 2\epsilon_{1xy}^{(1)} \\ \epsilon_{2xx}^{(1)} & \epsilon_{2yy}^{(1)} & \epsilon_{2zz}^{(1)} & 2\epsilon_{2yz}^{(1)} & 2\epsilon_{2xz}^{(1)} & 2\epsilon_{2xy}^{(1)} \\ \frac{\epsilon_{5xx}^{(1)} + \epsilon_{1xx}^{(2)}}{2} & \frac{\epsilon_{5yy}^{(1)} + \epsilon_{1yy}^{(2)}}{2} & \frac{\epsilon_{5zz}^{(1)} + \epsilon_{1zz}^{(2)}}{2} & \frac{2\epsilon_{5yz}^{(1)} + 2\epsilon_{1yz}^{(2)}}{2} & \frac{2\epsilon_{5xz}^{(1)} + 2\epsilon_{1xz}^{(2)}}{2} & \frac{2\epsilon_{5xy}^{(1)} + 2\epsilon_{1xy}^{(2)}}{2} \\ \frac{\epsilon_{6xx}^{(1)} + \epsilon_{2xx}^{(2)}}{2} & \frac{\epsilon_{6yy}^{(1)} + \epsilon_{2yy}^{(2)}}{2} & \frac{\epsilon_{6zz}^{(1)} + \epsilon_{2zz}^{(2)}}{2} & \frac{2\epsilon_{6yz}^{(1)} + 2\epsilon_{2yz}^{(2)}}{2} & \frac{2\epsilon_{6xz}^{(1)} + 2\epsilon_{2xz}^{(2)}}{2} & \frac{2\epsilon_{6xy}^{(1)} + 2\epsilon_{2xy}^{(2)}}{2} \\ \frac{\epsilon_{3xx}^{(1)} + \epsilon_{3xx}^{(2)}}{2} & \frac{\epsilon_{3yy}^{(1)} + \epsilon_{3yy}^{(2)}}{2} & \frac{\epsilon_{3zz}^{(1)} + \epsilon_{3zz}^{(2)}}{2} & \frac{2\epsilon_{3yz}^{(1)} + 2\epsilon_{3yz}^{(2)}}{2} & \frac{2\epsilon_{3xz}^{(1)} + 2\epsilon_{3xz}^{(2)}}{2} & \frac{2\epsilon_{3xy}^{(1)} + 2\epsilon_{3xy}^{(2)}}{2} \\ \frac{\epsilon_{4xx}^{(1)} + \epsilon_{4xx}^{(2)}}{2} & \frac{\epsilon_{4yy}^{(1)} + \epsilon_{4yy}^{(2)}}{2} & \frac{\epsilon_{4zz}^{(1)} + \epsilon_{4zz}^{(2)}}{2} & \frac{2\epsilon_{4yz}^{(1)} + 2\epsilon_{4yz}^{(2)}}{2} & \frac{2\epsilon_{4xz}^{(1)} + 2\epsilon_{4xz}^{(2)}}{2} & \frac{2\epsilon_{4xy}^{(1)} + 2\epsilon_{4xy}^{(2)}}{2} \\ \epsilon_{5xx}^{(2)} & \epsilon_{5yy}^{(2)} & \epsilon_{5zz}^{(2)} & 2\epsilon_{5yz}^{(2)} & 2\epsilon_{5xz}^{(2)} & 2\epsilon_{5xy}^{(2)} \\ \epsilon_{6xx}^{(2)} & \epsilon_{6yy}^{(2)} & \epsilon_{6zz}^{(2)} & 2\epsilon_{6yz}^{(2)} & 2\epsilon_{6xz}^{(2)} & 2\epsilon_{6xy}^{(2)} \end{pmatrix} \quad (2.87)$$

$$\begin{aligned}
\sigma_{raw} = & \left(\begin{array}{cccccc} \sigma_{1xx}^{(1)} & \sigma_{1yy}^{(1)} & \sigma_{1zz}^{(1)} & \sigma_{1yz}^{(1)} & \sigma_{1xz}^{(1)} & \sigma_{1xy}^{(1)} \\ \sigma_{2xx}^{(1)} & \sigma_{2yy}^{(1)} & \sigma_{2zz}^{(1)} & \sigma_{2yz}^{(1)} & \sigma_{2xz}^{(1)} & f_{2xy}^{(1)} \\ \sigma_{5xx}^{(1)} & \sigma_{5yy}^{(1)} & \sigma_{5zz}^{(1)} & \sigma_{5yz}^{(1)} & \sigma_{5xz}^{(1)} & \sigma_{5xy}^{(1)} \\ \sigma_{6xx}^{(1)} & \sigma_{6yy}^{(1)} & \sigma_{6zz}^{(1)} & \sigma_{6yz}^{(1)} & \sigma_{6xz}^{(1)} & \sigma_{6xy}^{(1)} \\ \sigma_{3xx}^{(1)} & \sigma_{3yy}^{(1)} & \sigma_{3zz}^{(1)} & \sigma_{3yz}^{(1)} & \sigma_{3xz}^{(1)} & \sigma_{3xy}^{(1)} \\ \sigma_{4xx}^{(1)} & \sigma_{4yy}^{(1)} & \sigma_{4zz}^{(1)} & \sigma_{4yz}^{(1)} & \sigma_{4xz}^{(1)} & \sigma_{4xy}^{(1)} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) + \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \sigma_{1xx}^{(2)} & \sigma_{1yy}^{(2)} & \sigma_{1zz}^{(2)} & \sigma_{1yz}^{(2)} & \sigma_{1xz}^{(2)} & \sigma_{1xy}^{(2)} \\ \sigma_{2xx}^{(2)} & \sigma_{2yy}^{(2)} & \sigma_{2zz}^{(2)} & \sigma_{2yz}^{(2)} & \sigma_{2xz}^{(2)} & \sigma_{2xy}^{(2)} \\ \sigma_{3xx}^{(2)} & \sigma_{3yy}^{(2)} & \sigma_{3zz}^{(2)} & \sigma_{3yz}^{(2)} & \sigma_{3xz}^{(2)} & \sigma_{3xy}^{(2)} \\ \sigma_{4xx}^{(2)} & \sigma_{4yy}^{(2)} & \sigma_{4zz}^{(2)} & \sigma_{4yz}^{(2)} & \sigma_{4xz}^{(2)} & \sigma_{4xy}^{(2)} \\ \sigma_{5xx}^{(2)} & \sigma_{5yy}^{(2)} & \sigma_{5zz}^{(2)} & \sigma_{5yz}^{(2)} & \sigma_{5xz}^{(2)} & \sigma_{5xy}^{(2)} \\ \sigma_{6xx}^{(2)} & \sigma_{6yy}^{(2)} & \sigma_{6zz}^{(2)} & \sigma_{6yz}^{(2)} & \sigma_{6xz}^{(2)} & \sigma_{6xy}^{(2)} \end{array} \right) \\
\sigma = & \left(\begin{array}{cccccc} \sigma_{1xx}^{(1)} & \sigma_{1yy}^{(1)} & \sigma_{1zz}^{(1)} & \sigma_{1yz}^{(1)} & \sigma_{1xz}^{(1)} & \sigma_{1xy}^{(1)} \\ \sigma_{2xx}^{(1)} & \sigma_{2yy}^{(1)} & \sigma_{2zz}^{(1)} & \sigma_{2yz}^{(1)} & \sigma_{2xz}^{(1)} & f_{2xy}^{(1)} \\ \frac{\sigma_{5xx}^{(1)} + \sigma_{1xx}^{(2)}}{2} & \frac{\sigma_{5yy}^{(1)} + \sigma_{1yy}^{(2)}}{2} & \frac{\sigma_{5zz}^{(1)} + \sigma_{1zz}^{(2)}}{2} & \frac{\sigma_{5yz}^{(1)} + \sigma_{1yz}^{(2)}}{2} & \frac{\sigma_{5xz}^{(1)} + \sigma_{1xz}^{(2)}}{2} & \frac{\sigma_{5xy}^{(1)} + \sigma_{1xy}^{(2)}}{2} \\ \frac{\sigma_{6xx}^{(1)} + \sigma_{2xx}^{(2)}}{2} & \frac{\sigma_{6yy}^{(1)} + \sigma_{2yy}^{(2)}}{2} & \frac{\sigma_{6zz}^{(1)} + \sigma_{2zz}^{(2)}}{2} & \frac{\sigma_{6yz}^{(1)} + \sigma_{2yz}^{(2)}}{2} & \frac{\sigma_{6xz}^{(1)} + \sigma_{2xz}^{(2)}}{2} & \frac{\sigma_{6xy}^{(1)} + \sigma_{2xy}^{(2)}}{2} \\ \frac{\sigma_{3xx}^{(1)} + \sigma_{3xx}^{(2)}}{2} & \frac{\sigma_{3yy}^{(1)} + \sigma_{3yy}^{(2)}}{2} & \frac{\sigma_{3zz}^{(1)} + \sigma_{3zz}^{(2)}}{2} & \frac{\sigma_{3yz}^{(1)} + \sigma_{3yz}^{(2)}}{2} & \frac{\sigma_{3xz}^{(1)} + \sigma_{3xz}^{(2)}}{2} & \frac{\sigma_{3xy}^{(1)} + \sigma_{3xy}^{(2)}}{2} \\ \frac{\sigma_{4xx}^{(1)} + \sigma_{4xx}^{(2)}}{2} & \frac{\sigma_{4yy}^{(1)} + \sigma_{4yy}^{(2)}}{2} & \frac{\sigma_{4zz}^{(1)} + \sigma_{4zz}^{(2)}}{2} & \frac{\sigma_{4yz}^{(1)} + \sigma_{4yz}^{(2)}}{2} & \frac{\sigma_{4xz}^{(1)} + \sigma_{4xz}^{(2)}}{2} & \frac{\sigma_{4xy}^{(1)} + \sigma_{4xy}^{(2)}}{2} \\ \sigma_{5xx}^{(2)} & \sigma_{5yy}^{(2)} & \sigma_{5zz}^{(2)} & \sigma_{5yz}^{(2)} & \sigma_{5xz}^{(2)} & \sigma_{5xy}^{(2)} \\ \sigma_{6xx}^{(2)} & \sigma_{6yy}^{(2)} & \sigma_{6zz}^{(2)} & \sigma_{6yz}^{(2)} & \sigma_{6xz}^{(2)} & \sigma_{6xy}^{(2)} \end{array} \right) \quad (2.88)
\end{aligned}$$

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). 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. 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. 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 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.

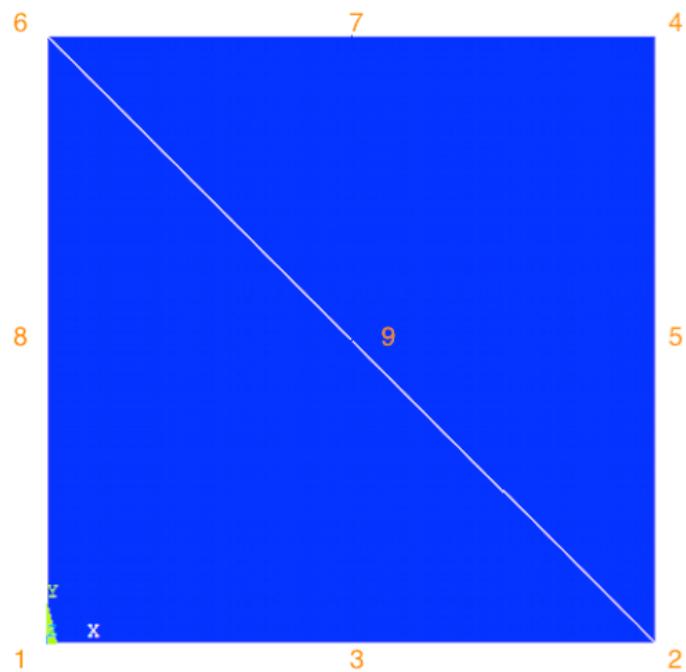


Figure 3.1: A simple meshed geometry

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						
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

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 reason for failed test. The element types to be tested are: Tri3, Tri6, Quad4, Quad8, Tet4 and Tet10. It is of interest to test strain and stress components. 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 component 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 type in AMfe Toolbox. As a comparison, we need to define the absolute error and relative error in the test. Absolute error is the magnitude of the difference between actual value and approximate value. It can be defined as:

$$e_{abs} = \|X_{actual} - X_{fem}\| \quad (4.1)$$

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

$$e_{rel} = \frac{\|X_{actual} - X_{fem}\|}{X_{actual}} \quad (4.2)$$

It is noteworthy that errors of approximation or rounding and truncation are introduced by using numerical methods or algorithms and computing with finite precision. Truncation error describes the errors, which occurs because some series (finite or infinite) is truncated to a fewer number of terms. Such errors are virtually algorithmic errors. Roundoff error occurs because of the computing tool's inability to handle certain numbers [Dukkipati 2010]. 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 to set the tolerance with different values and evaluate the highest tolerance for each element type. The complete test of all element types is shown in Table 4.1. It is obvious that only Tri3 shows a good performance and Quad4 has a relative good quality. It is necessary to investigate into what is wrong with the other element types.

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

4.2 The Patch Test

After testing the quality of each element type, we need to know if the finite element program has the correct algorithmic procedure. It 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 to maintain a constant strain/stress field. This is to simplify a complex problem to a simple one, to which we already know the correct numerical results.

The procedure of the displacement patch test is as follows: create a simple geometry that consist of one or several element; pick a patch and apply the displacement at exterior nodes and the prescribed displacement field will be set as $u_x = x, u_y = 0$. The reason that the displacement field only applies at 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. 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 4.1. Strain field can be viewed in Figure 4.2 from ANSYS. As expected, $\epsilon_{xx} = 0$. Thus, we have built the model for a patch test successfully. Now it is time to check if AMfe Toolbox has the same constant strain field. We export the node, element, and displacement data from ANSYS and import all of them into AMfe Toolbox. After computing of strain and stress in the AMfe, the contour plot of strain is shown in ParaView as the right side of Figure 4.2. The plot from AMfe Toolbox illustrates the same result $\epsilon_{xx} = 0$. It proves that the displacement patch test has been passed. It is worth noting that the patch test has only two result: success or failure. There is no in-between state like "passing the patch test by 90 percent." Another element types namely Tri3, Quad4, Quad8, Tet4, and Tet10 have also passed the displacement patch test. Taking into account the limited space of this thesis, we are not listing the details here. From the patch test we can not find the reason why Quad8, Tri6, Tet4 and Tet10 did not passed the unit test. This is still an issue to discuss later.

4.3 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 using each element type. The main motivation of complex model test is to compare the strain and stress contour plot of each element type from ANSYS and AMfe Toolbox. Furthermore, the comparison between

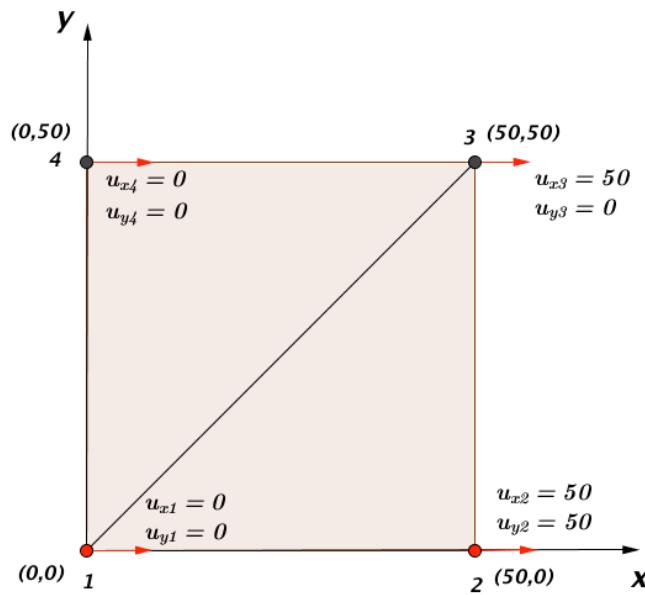


Figure 4.1: The displacement patch test with Tri6 element

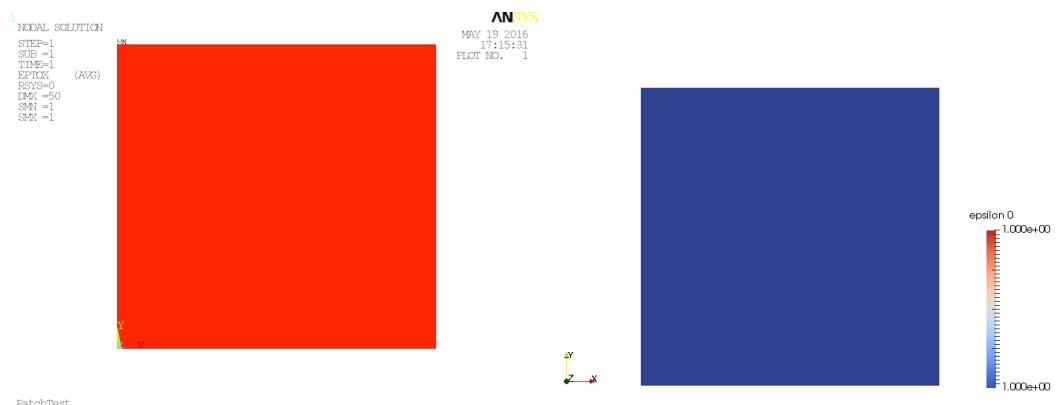


Figure 4.2: left: Contour plot of ϵ_{xx} from ANSYS (SMN: minimal value; SMX: maximal value); right: Contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

analogous elements (Quad4 and Quad8; Tri3 and Tri6; Tet4 and Tet10) are also significant for our research. Figure 4.3 - Figure 4.8 illustrate the contour plot of strain and stress in all direction, which is meshed with.. In a sum, all of these element type in AMfe Toolbox shows the exact contour plot in compare with the ANSYS generating contour plot. For the bound of value, Tri3 and Quad4 in AMfe Toolbox have almost the same value as them in ANSYS. The other element type have more or less problems.

The relation between the numbering of components in ParaView and the tensor direction follows this rule: 0-xx, 1-yy, 2-zz, 3-yx, 4-yz, 5-xz, 6-zz, 7-xz, 8-xy.

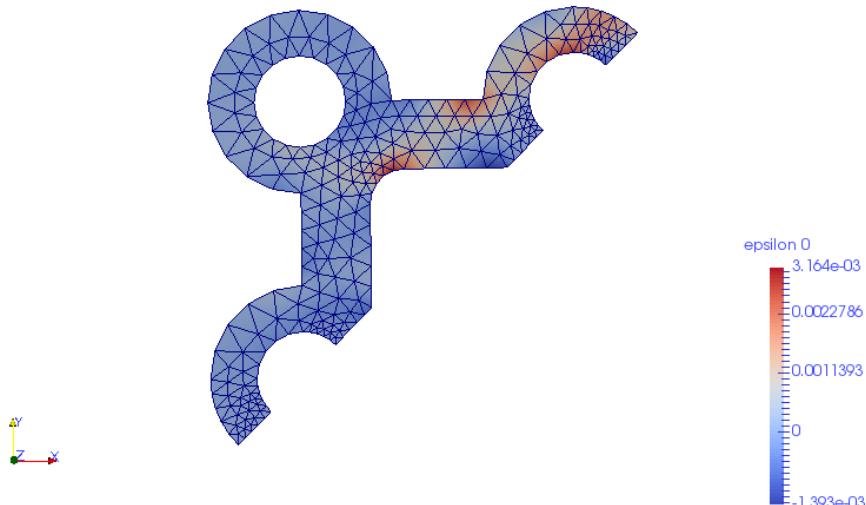
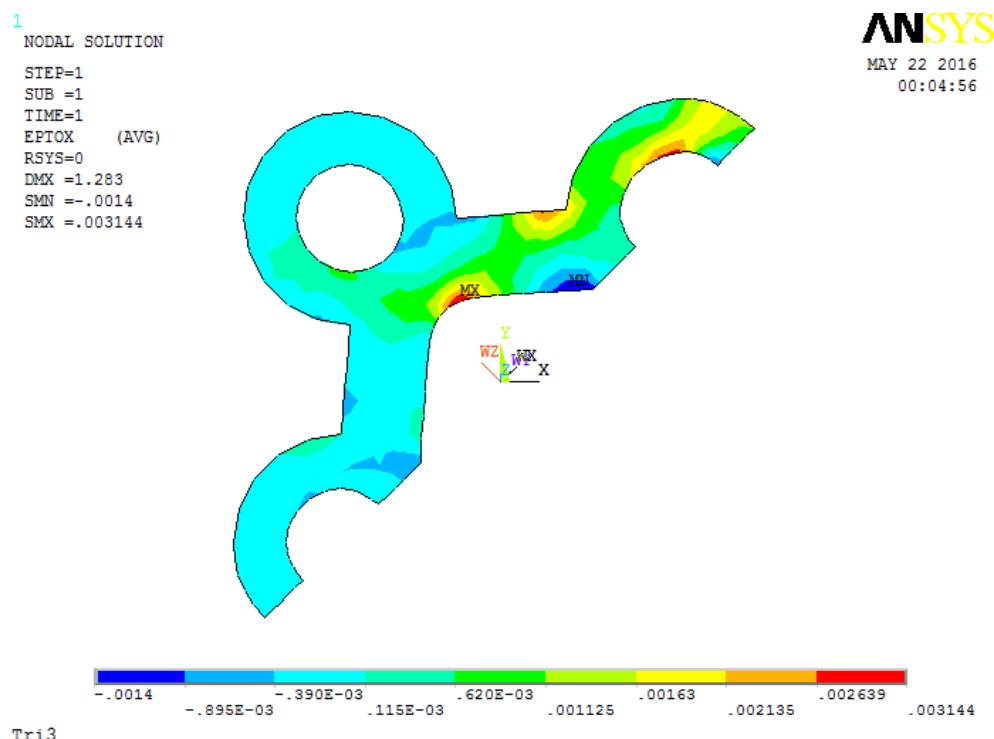


Figure 4.3: Mesh with Tri3. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

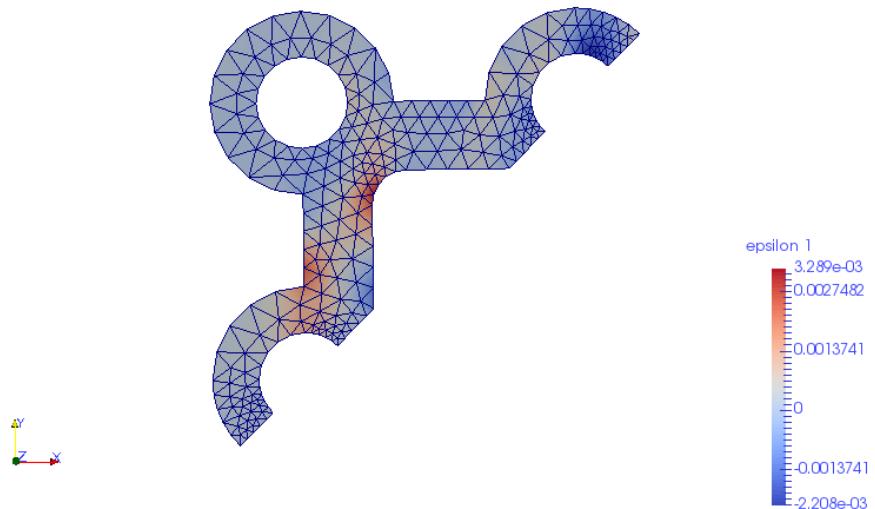
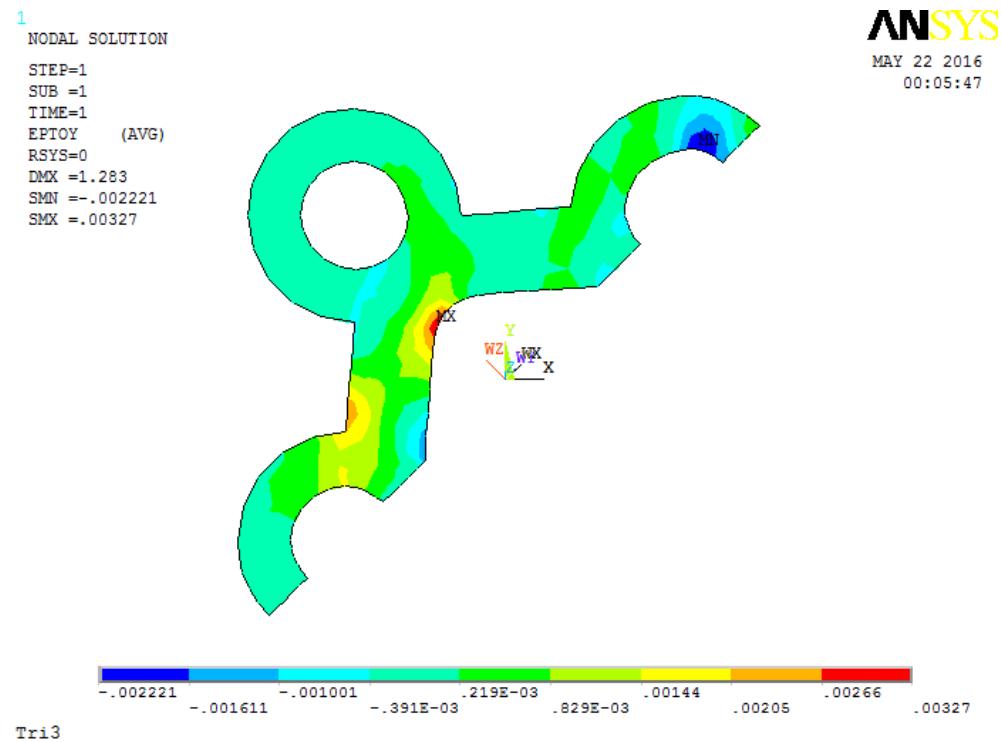


Figure 4.4: Mesh with Tri3. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

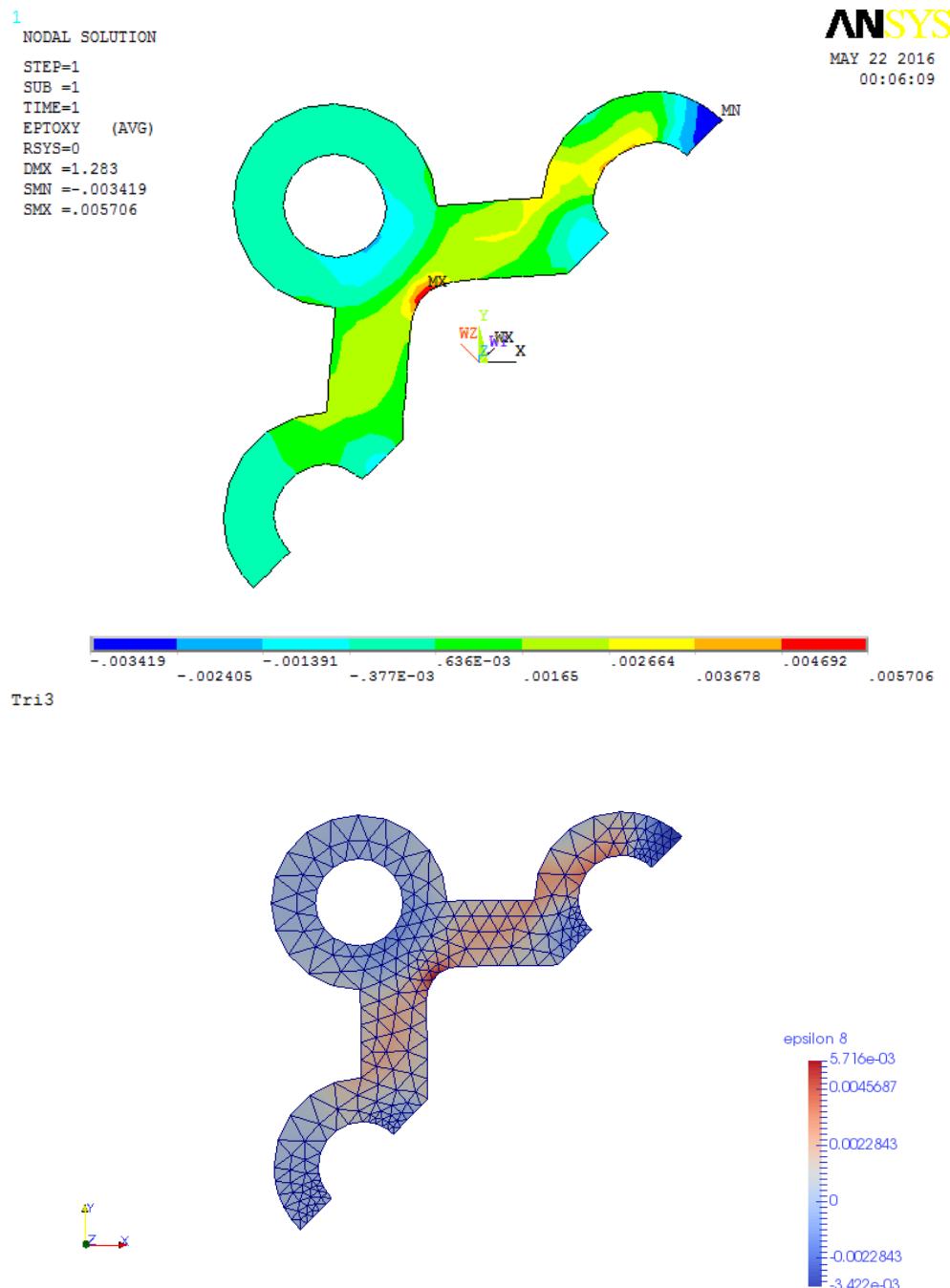


Figure 4.5: Mesh with Tri3. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

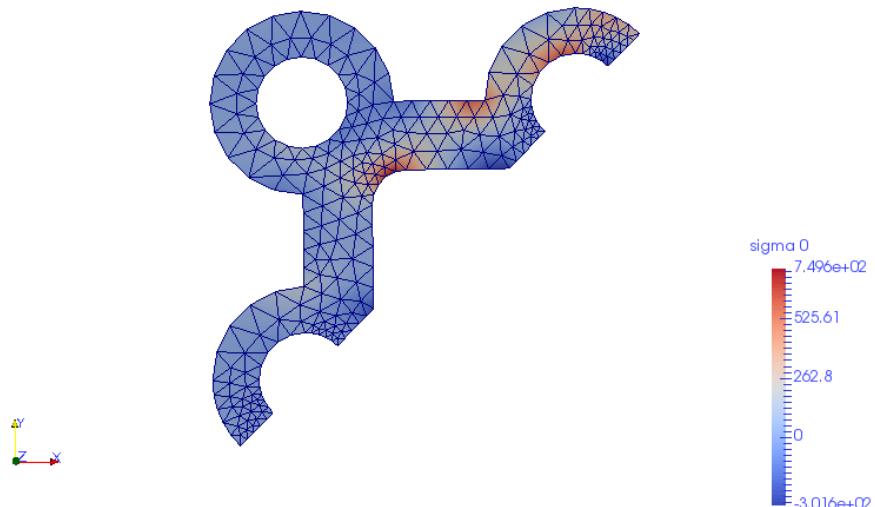
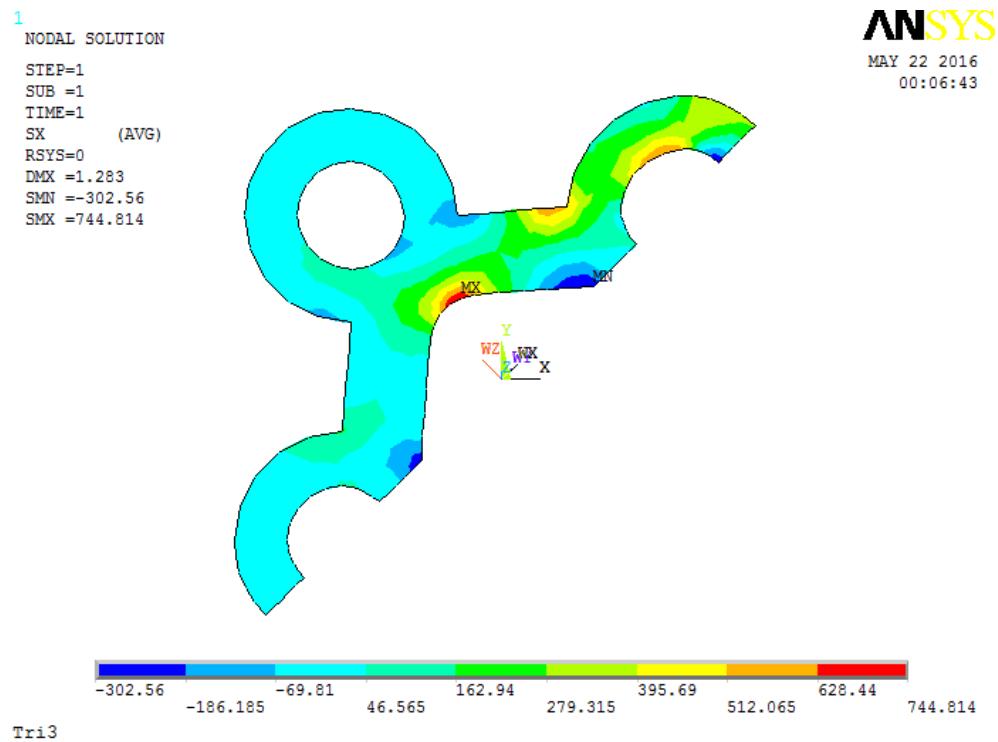


Figure 4.6: Mesh with Tri3. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfE, demonstrate in ParaView

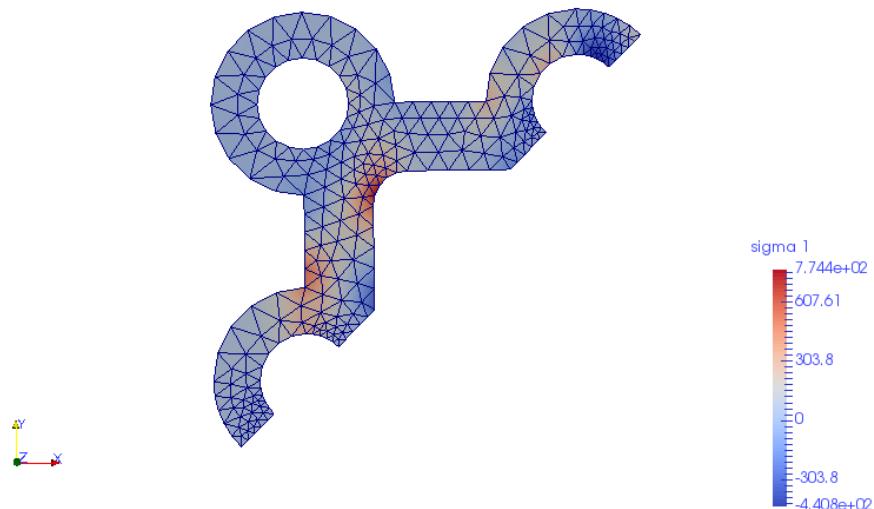
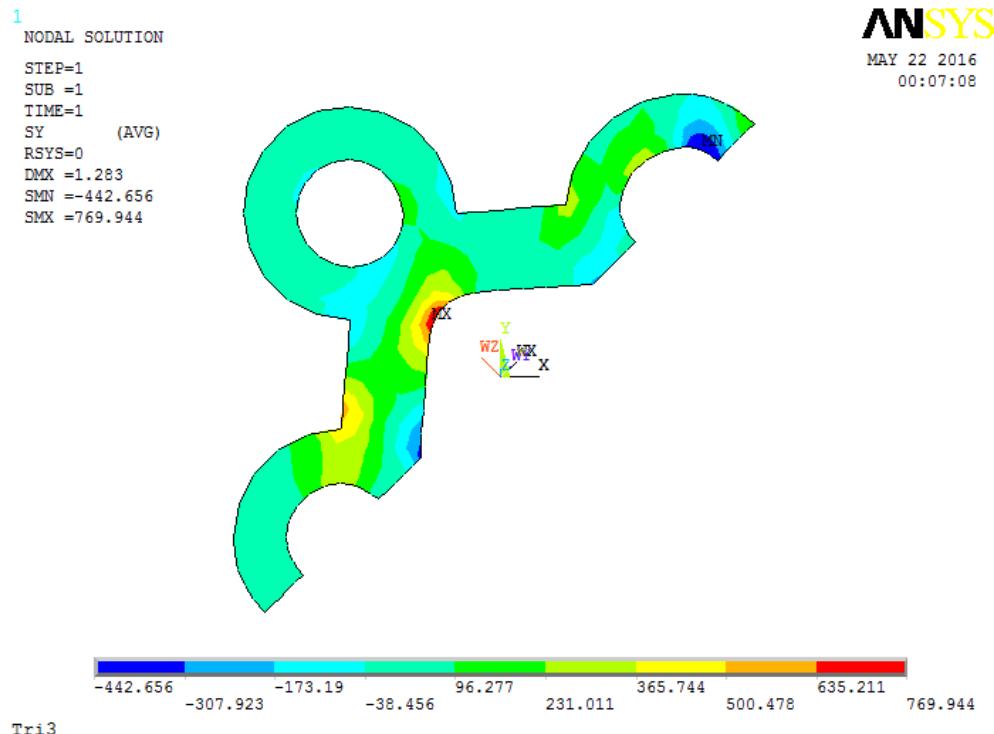


Figure 4.7: Mesh with Tri3. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

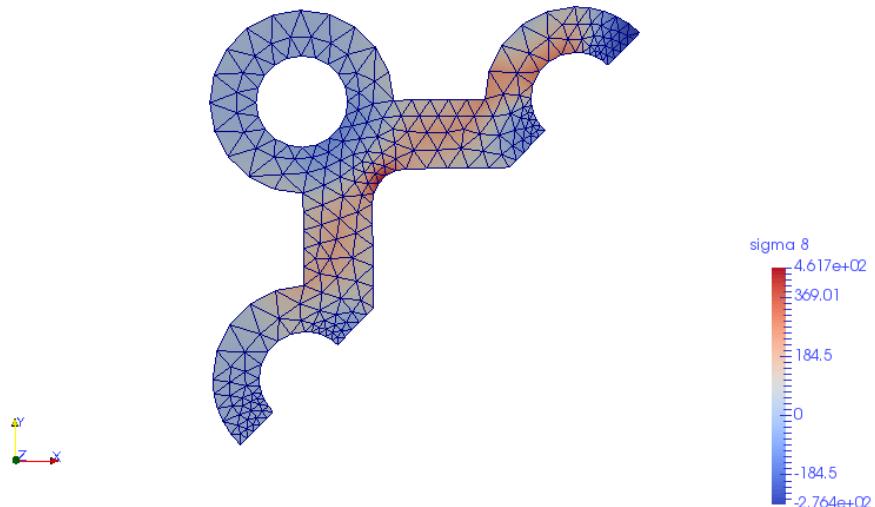
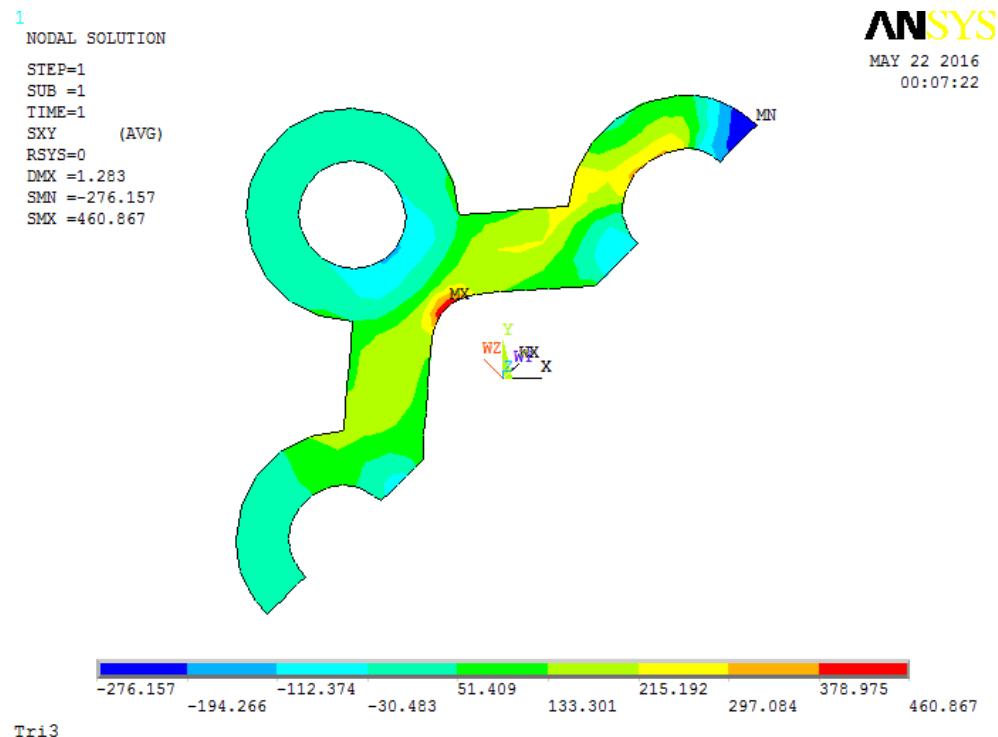


Figure 4.8: Mesh with Tri3. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

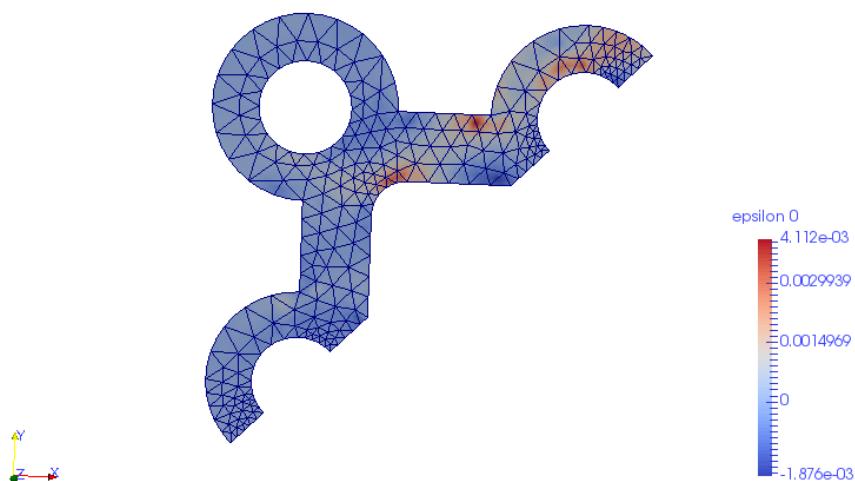
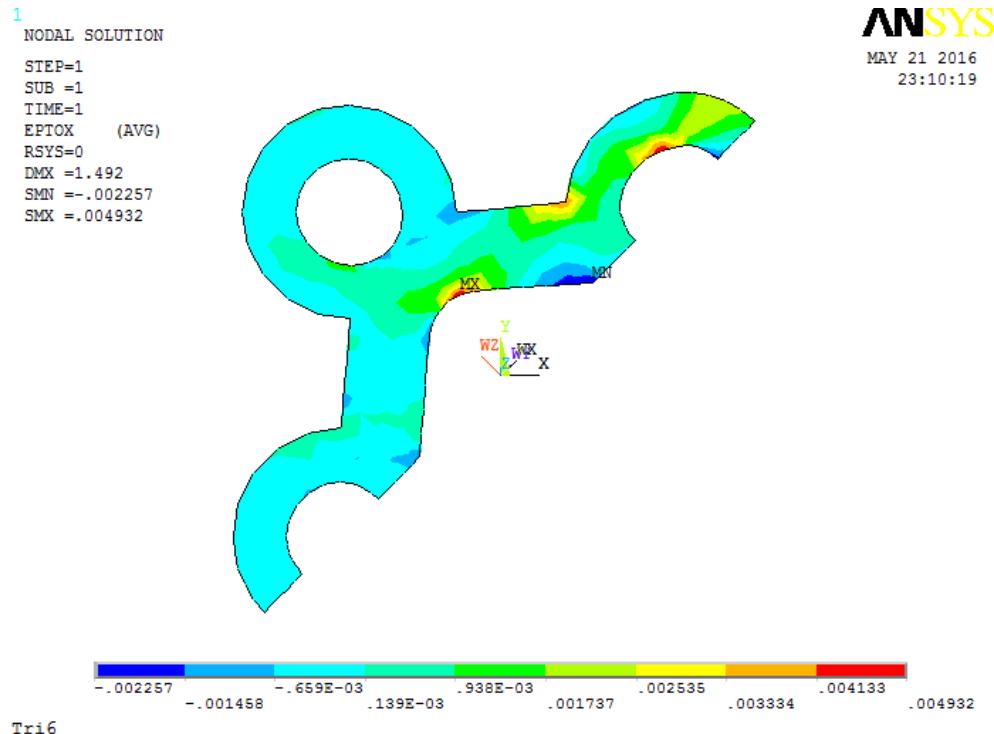


Figure 4.9: Mesh with Tri6. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

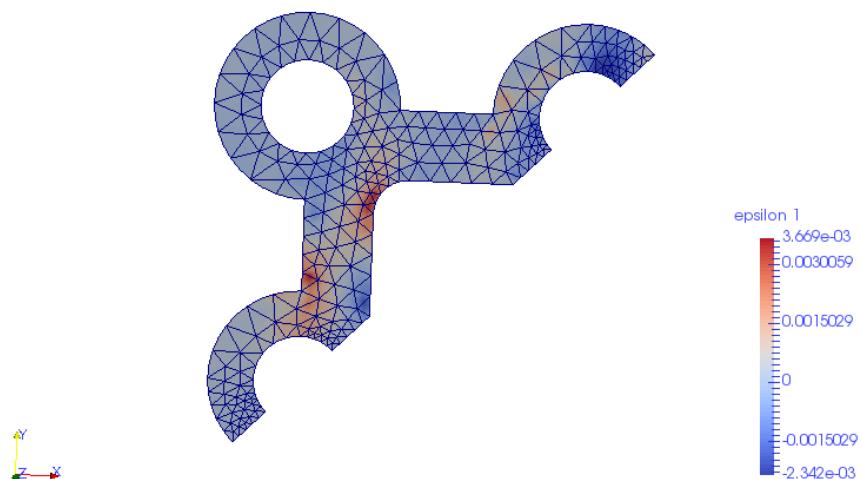
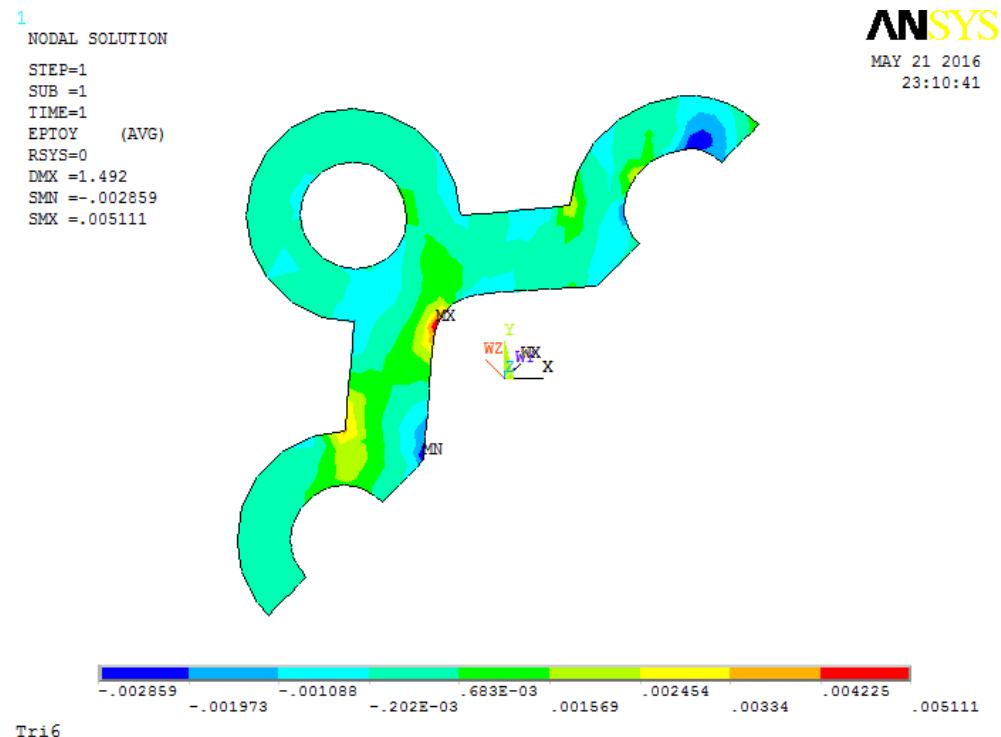


Figure 4.10: Mesh with Tri6. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

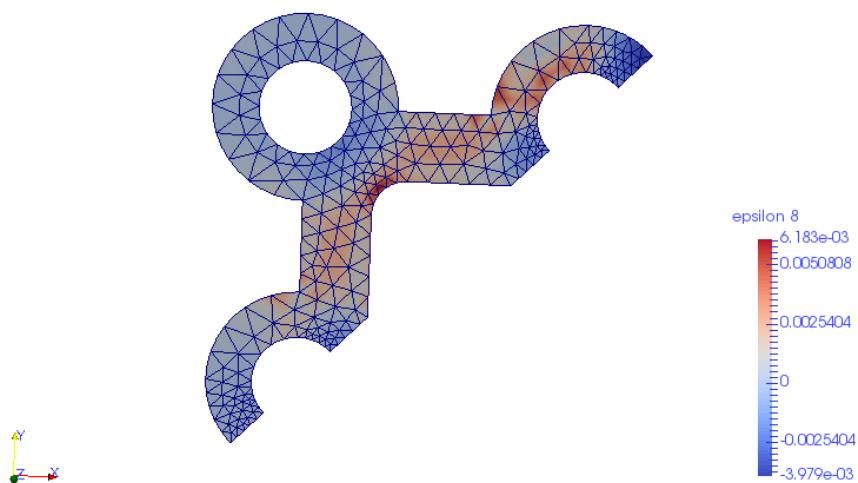
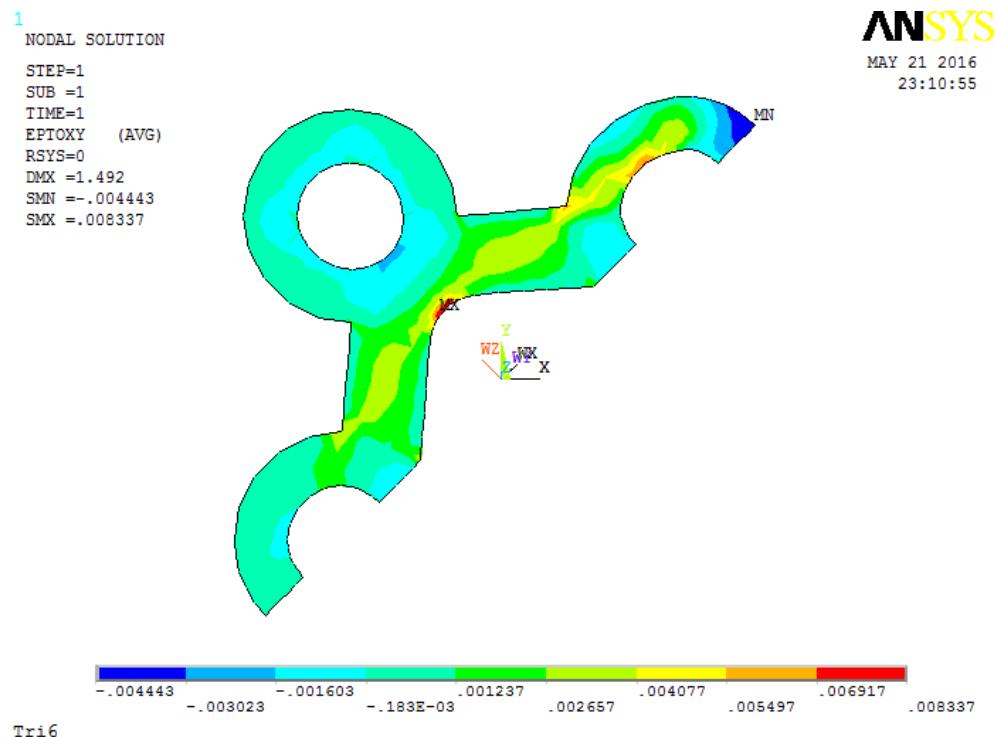


Figure 4.11: Mesh with Tri6. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

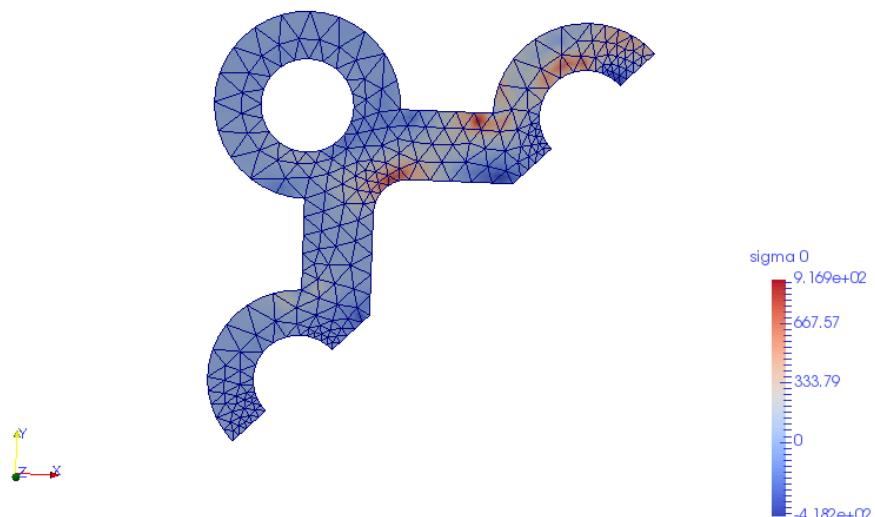
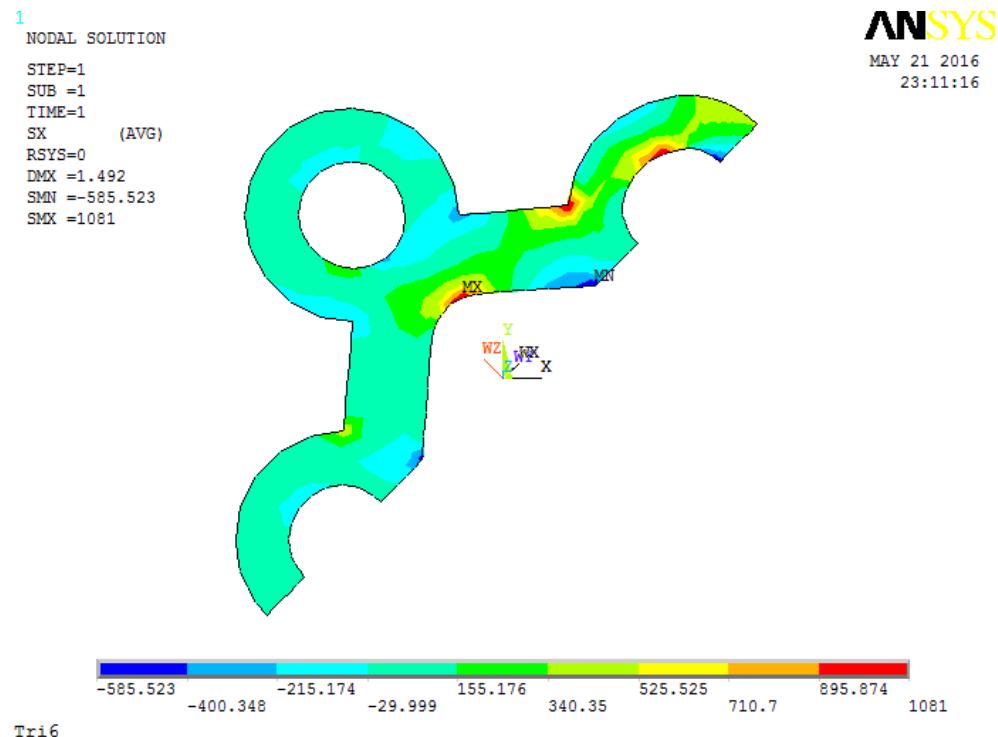


Figure 4.12: Mesh with Tri6. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView

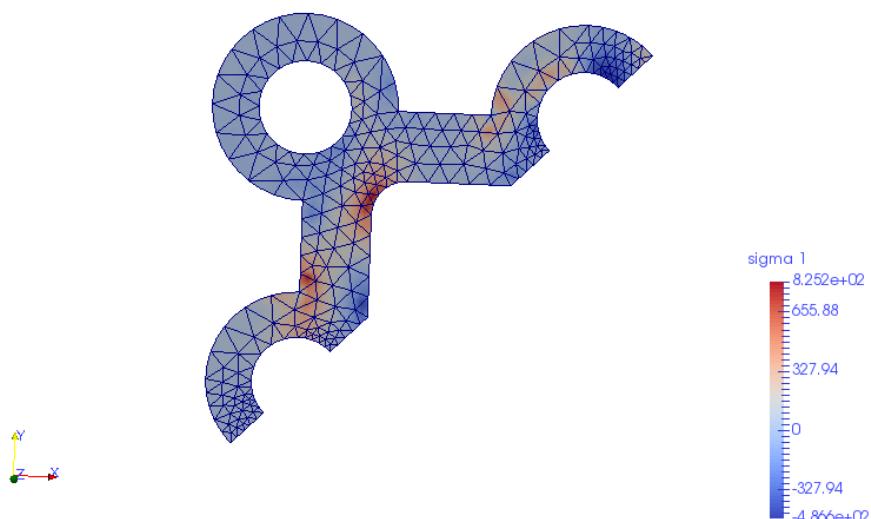
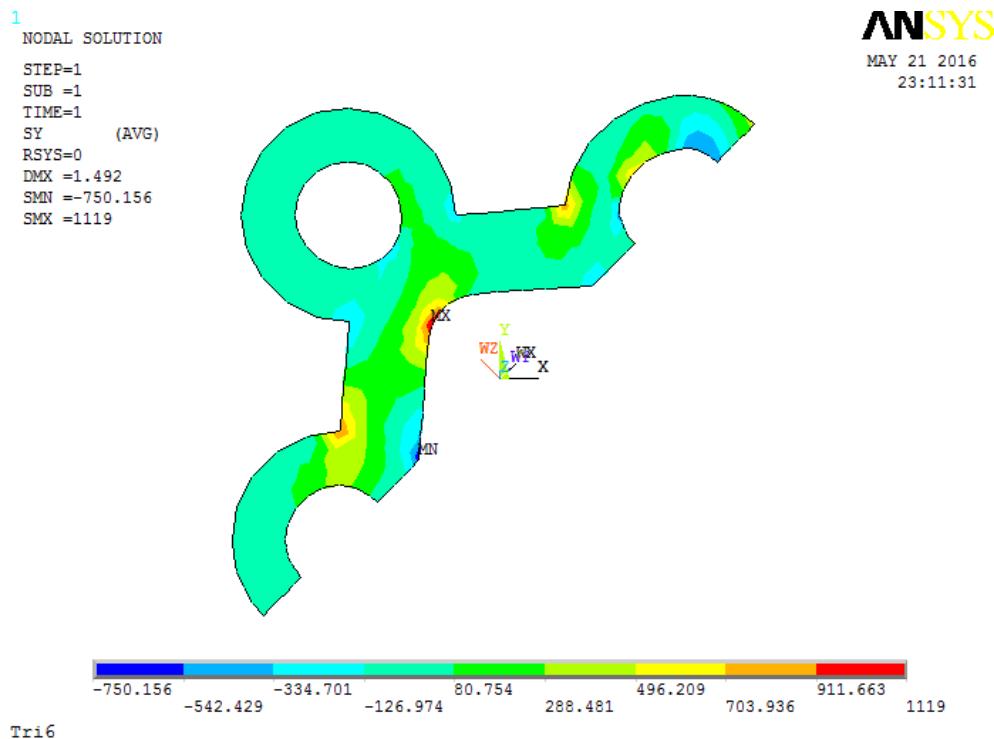


Figure 4.13: Mesh with Tri6. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

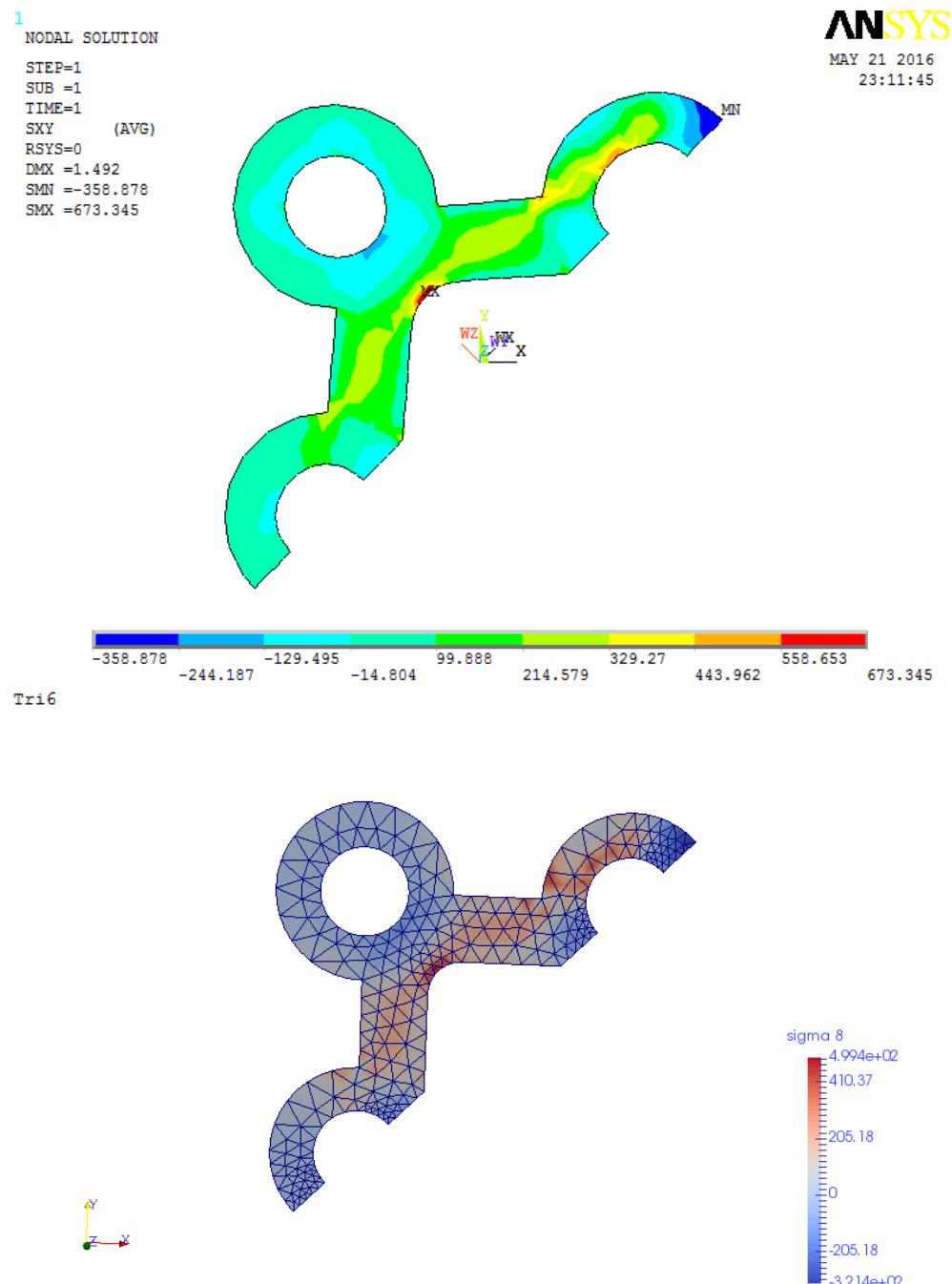


Figure 4.14: Mesh with Tri6. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

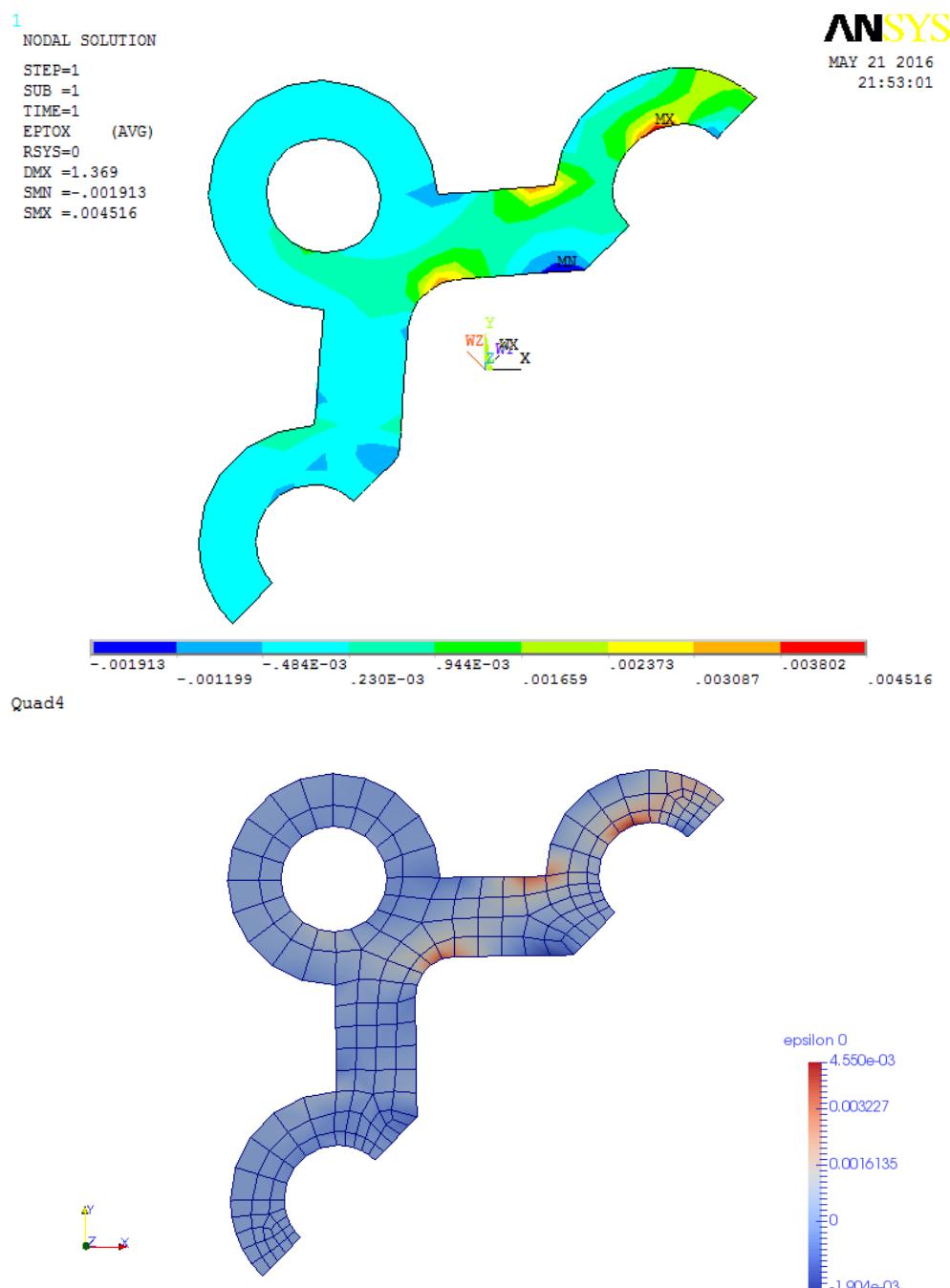


Figure 4.15: Mesh with Quad4. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

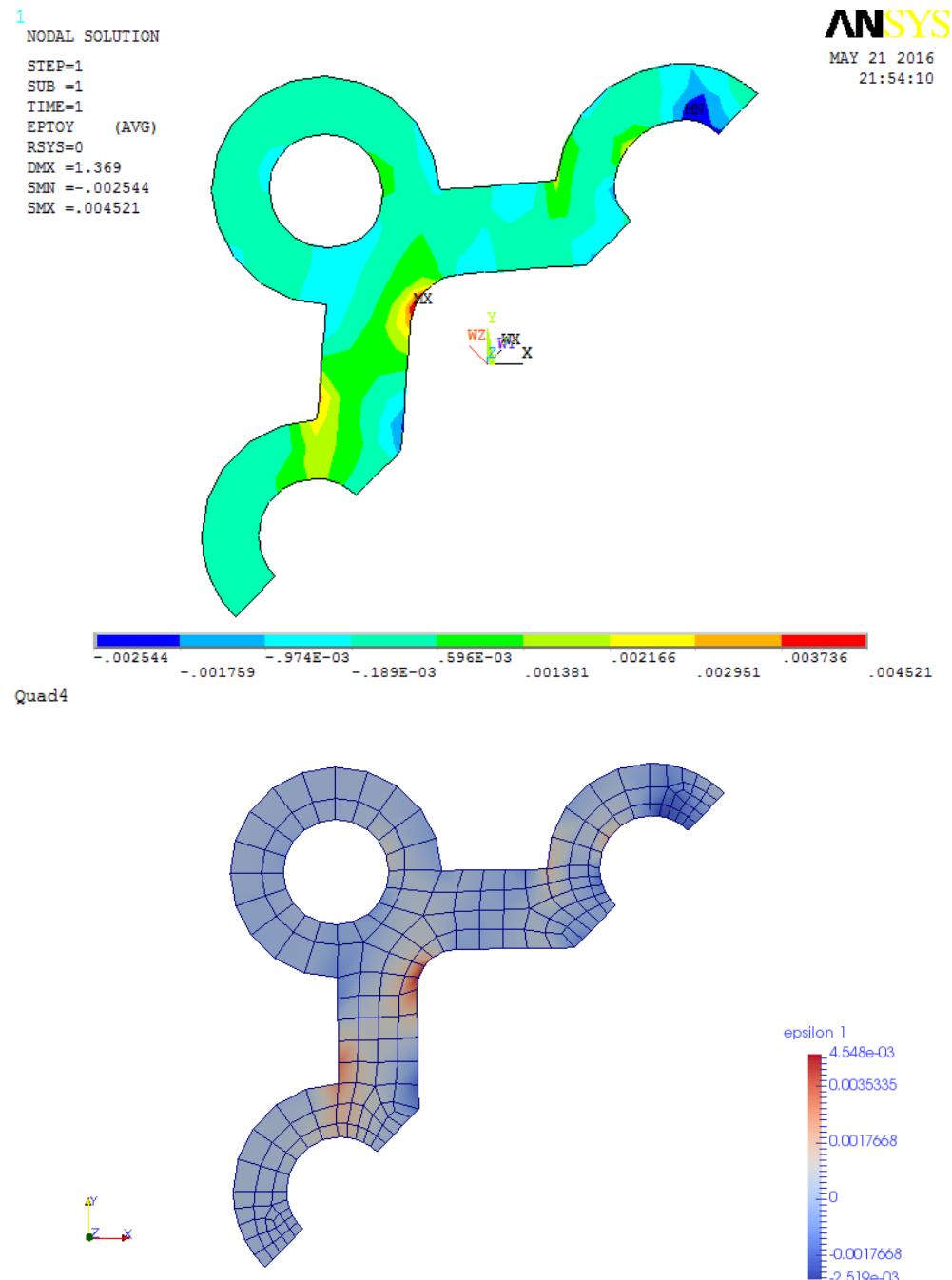


Figure 4.16: Mesh with Quad4. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

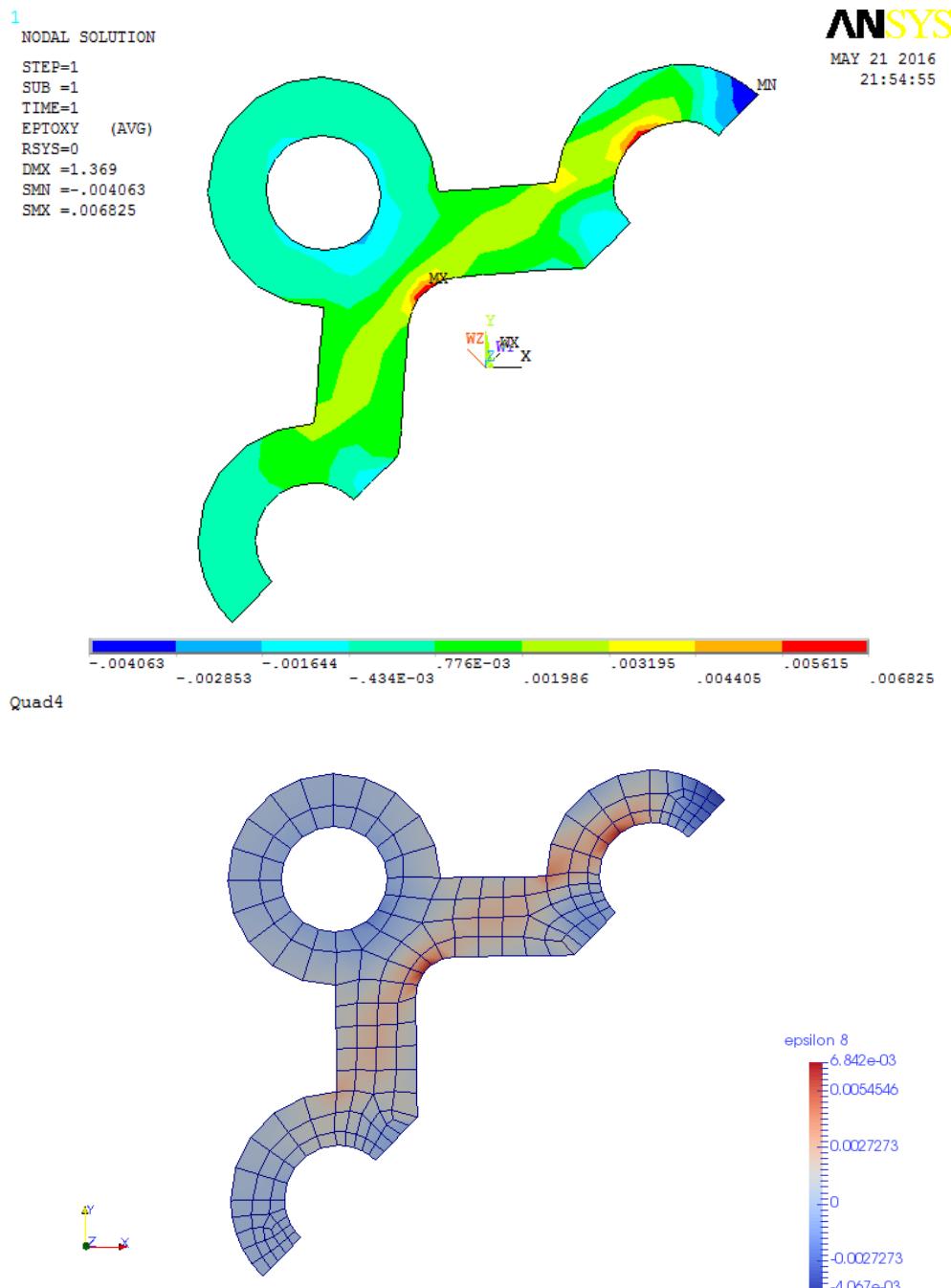


Figure 4.17: Mesh with Quad4. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

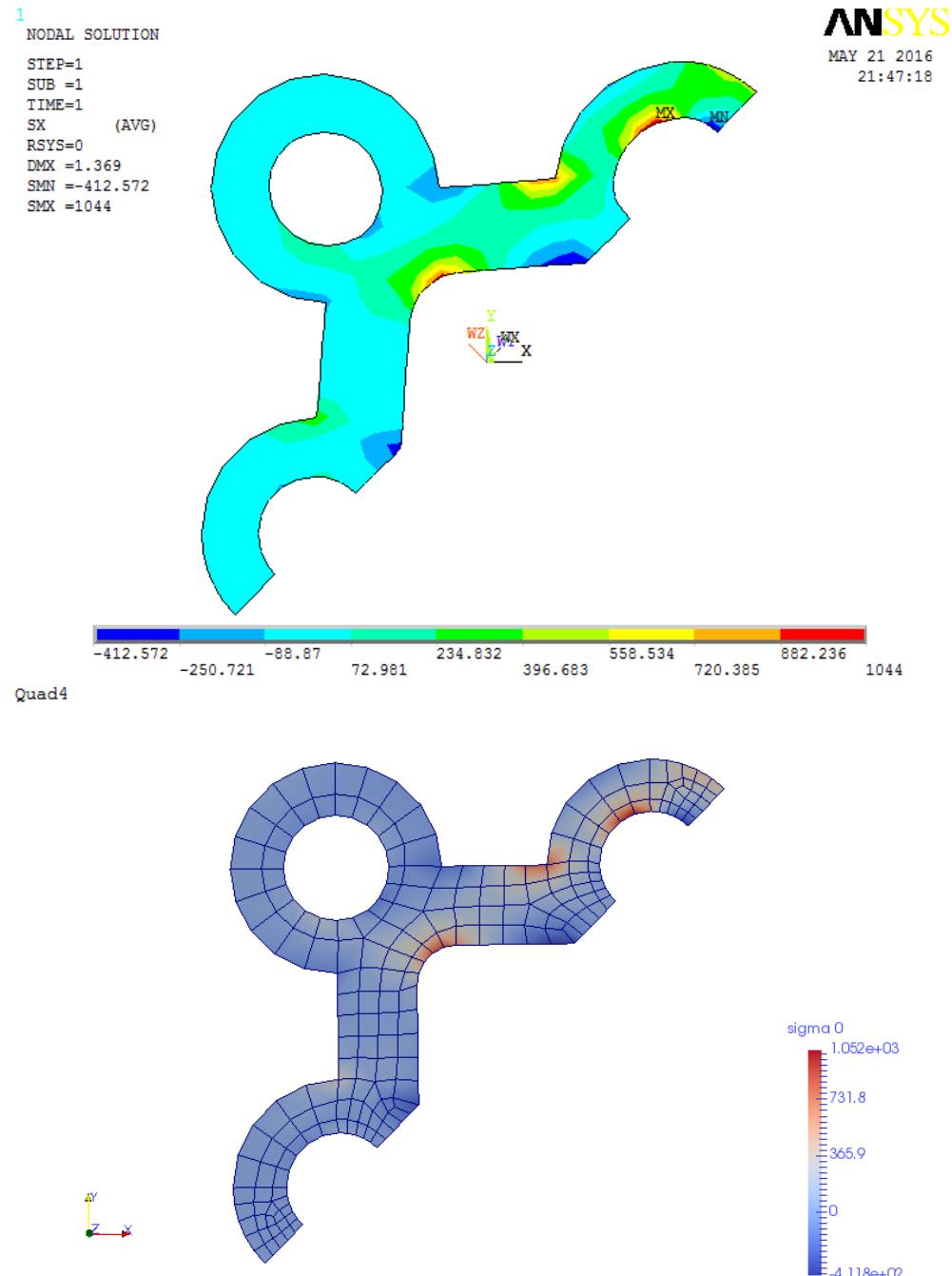


Figure 4.18: Mesh with Quad4. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView

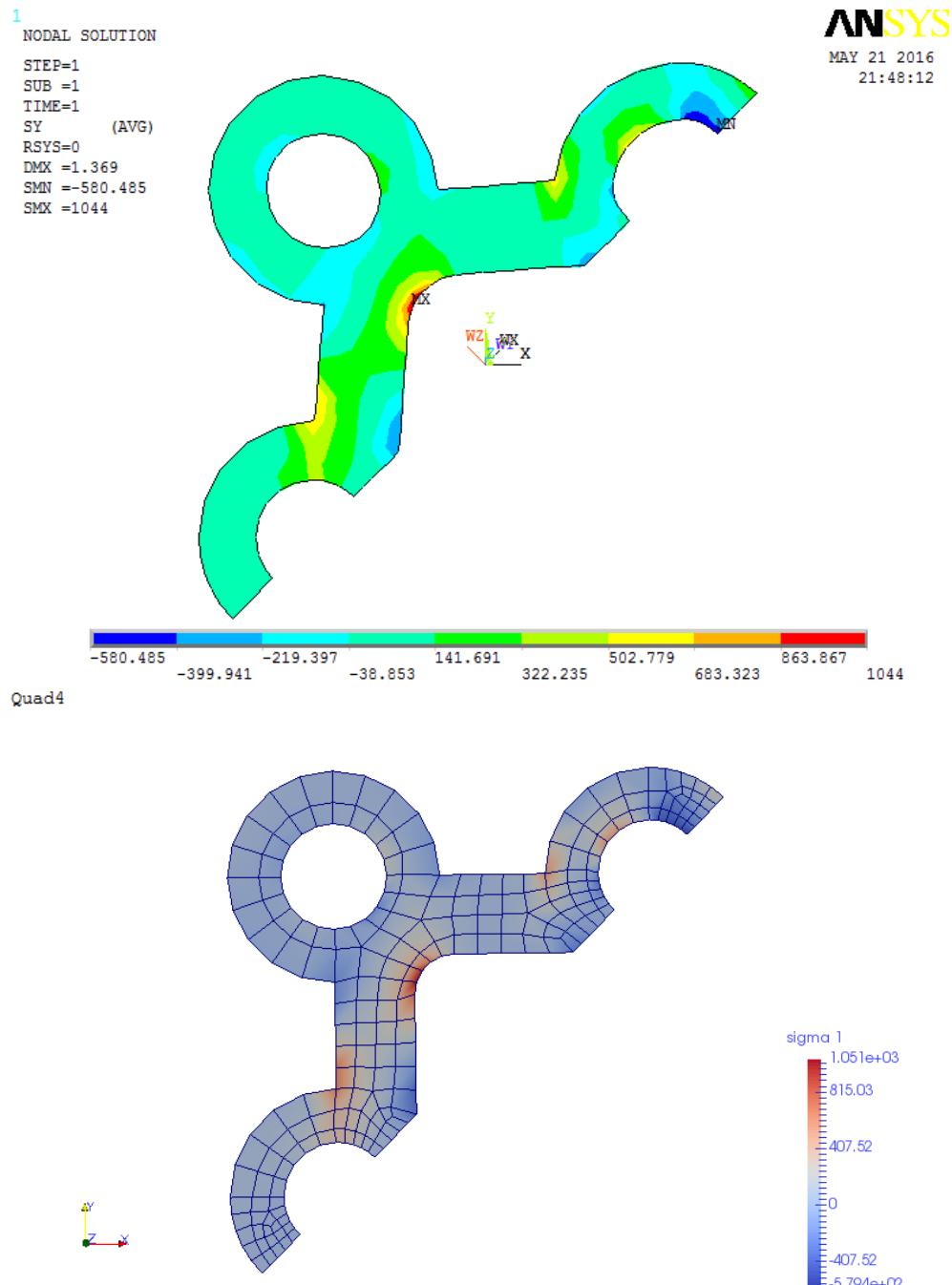


Figure 4.19: Mesh with Quad4. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

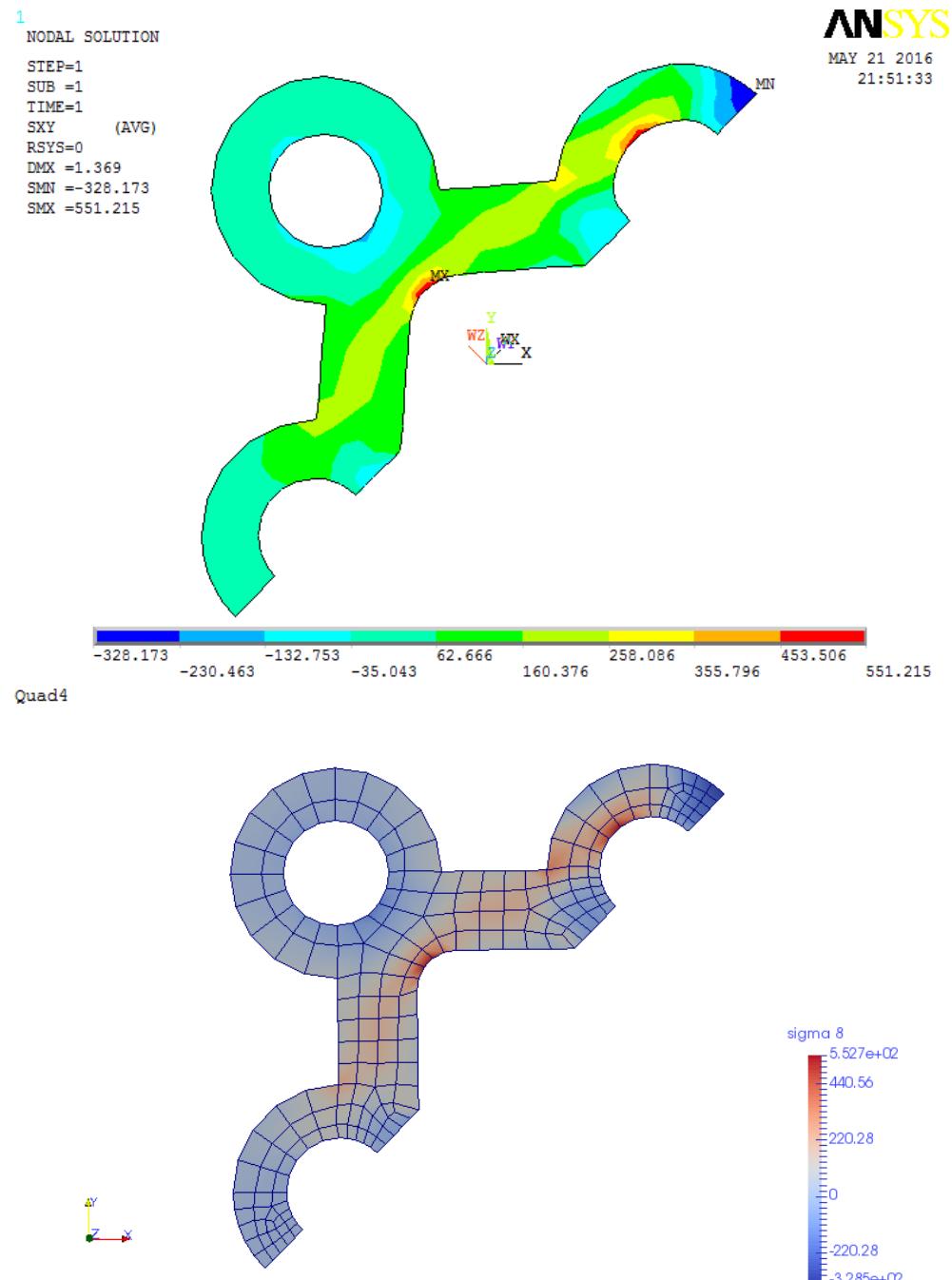


Figure 4.20: Mesh with Quad4. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

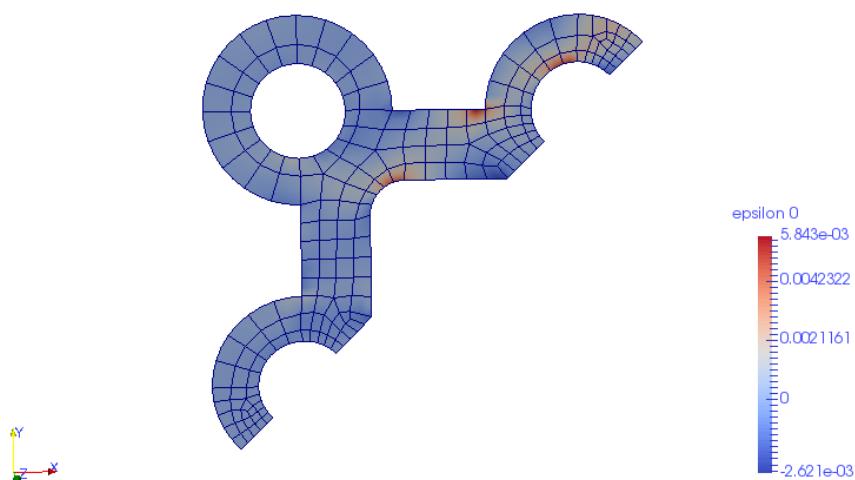
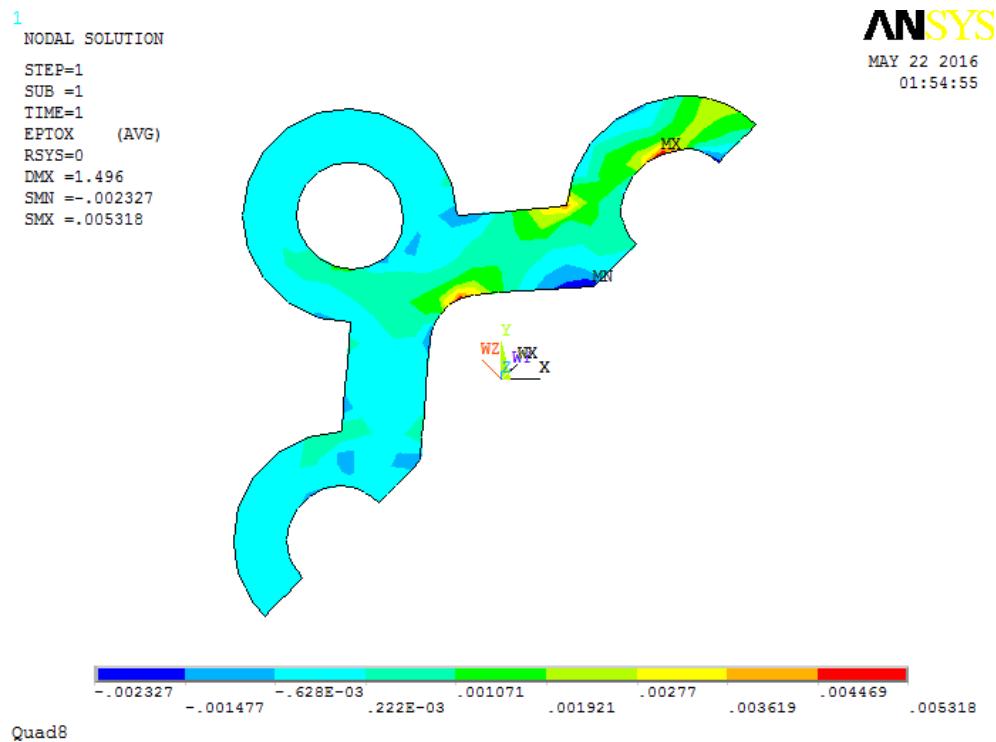


Figure 4.21: Mesh with Quad8. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

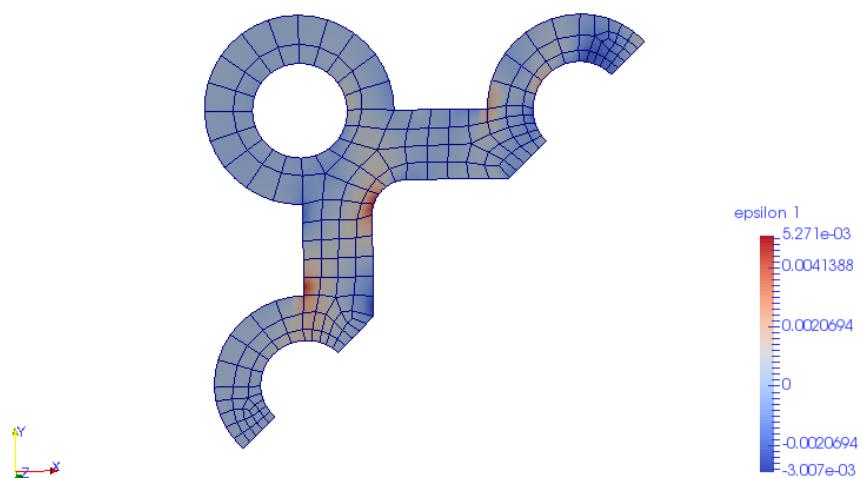
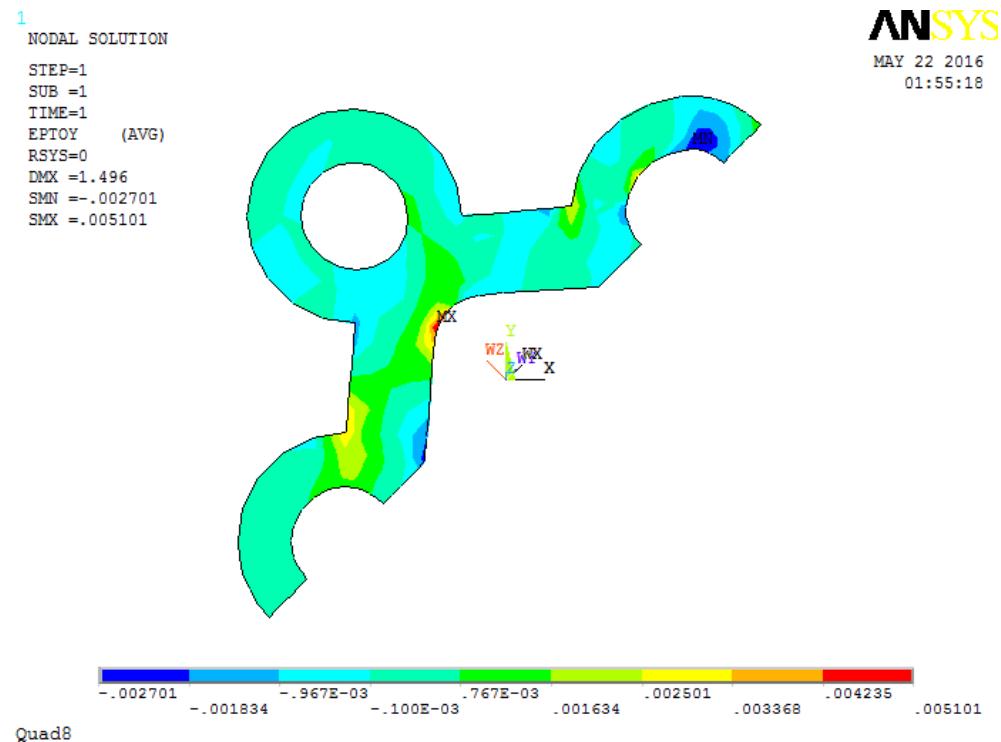


Figure 4.22: Mesh with Quad8. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

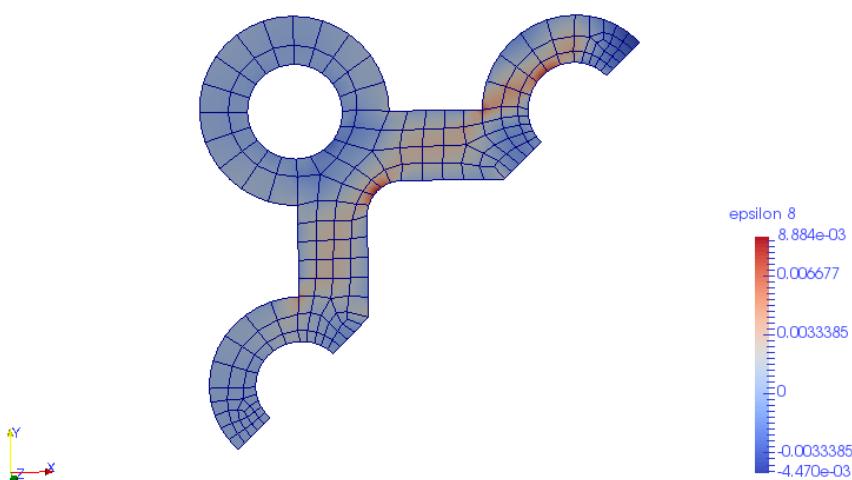
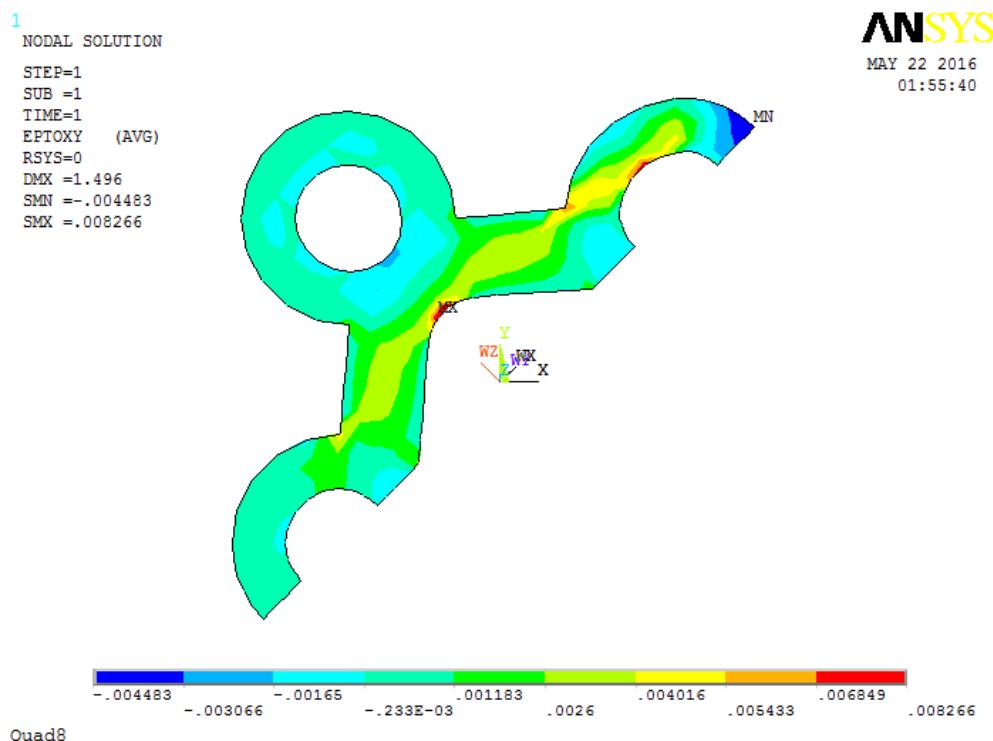


Figure 4.23: Mesh with Quad8. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

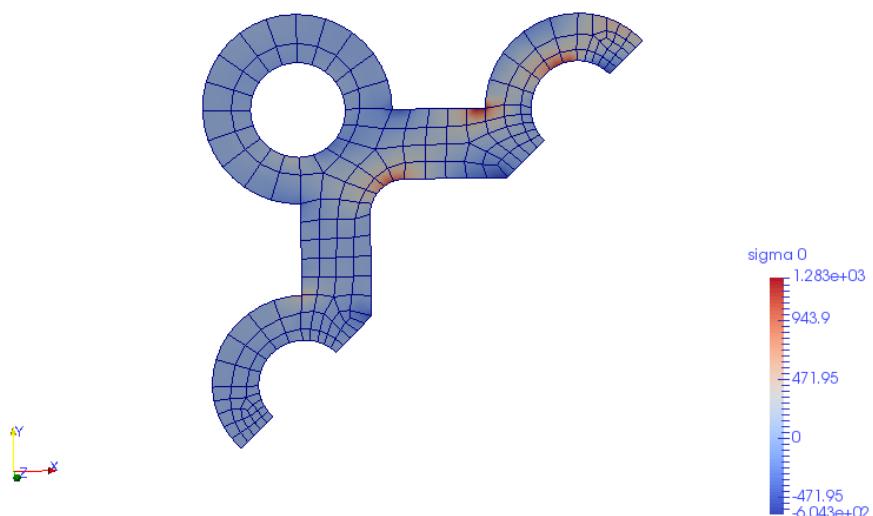
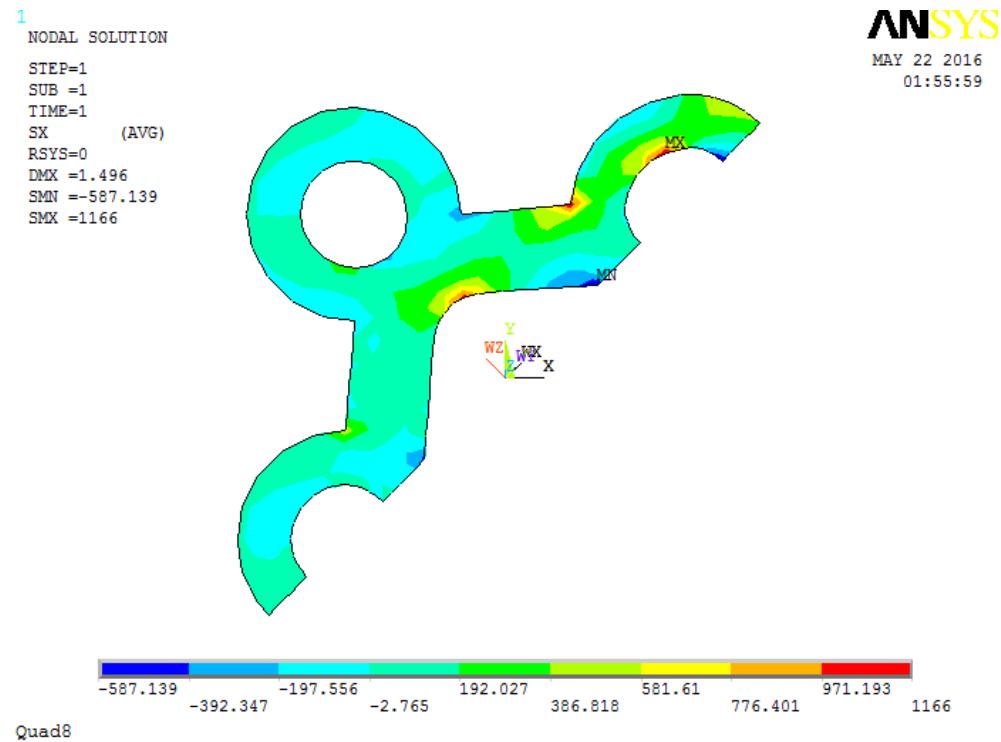


Figure 4.24: Mesh with Quad8. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView

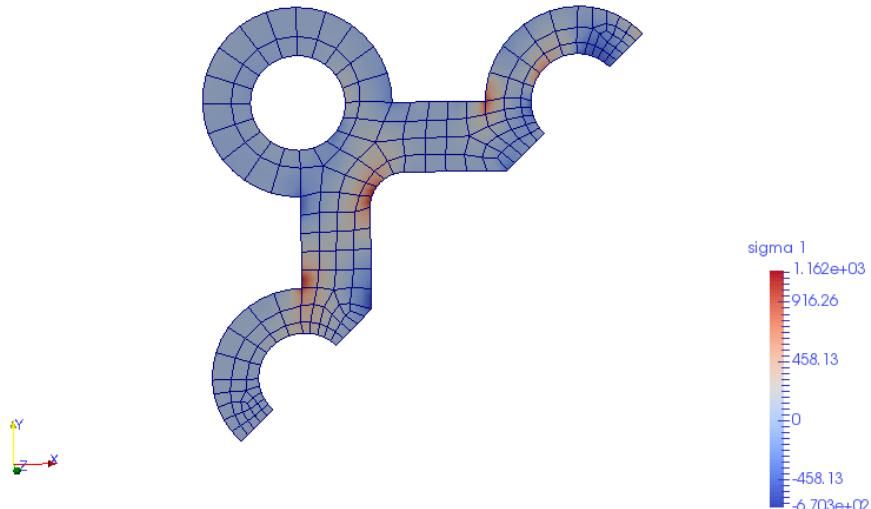
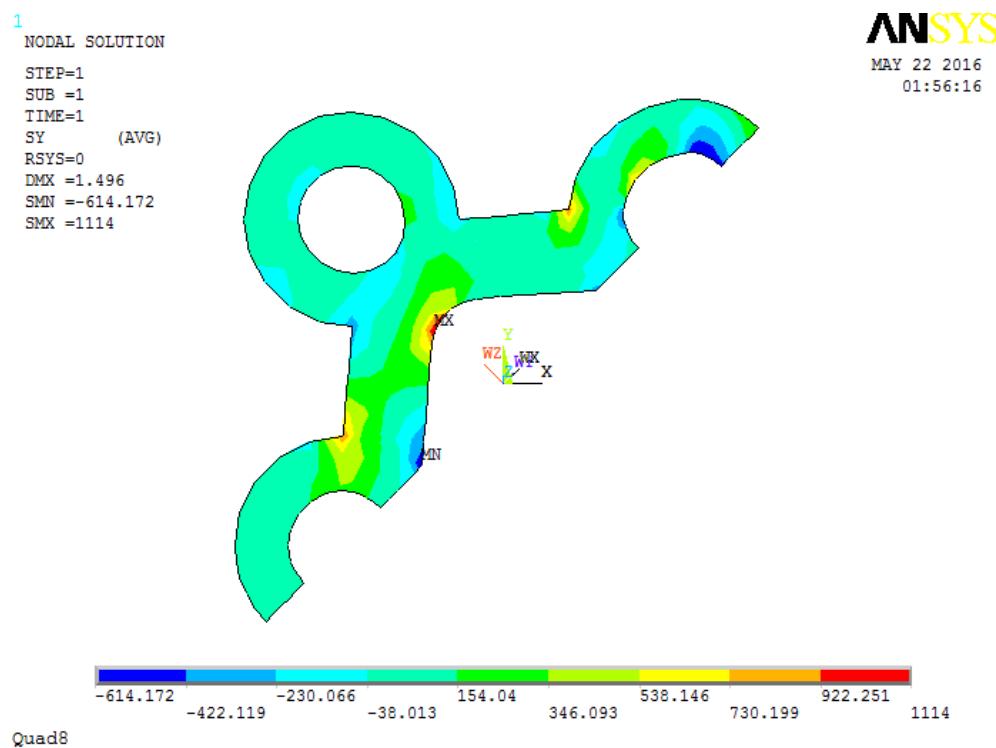


Figure 4.25: Mesh with Quad8. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

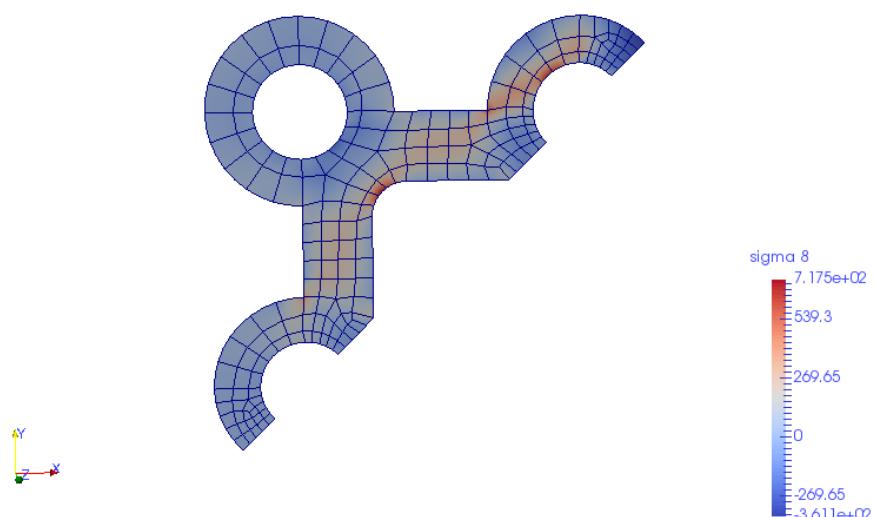
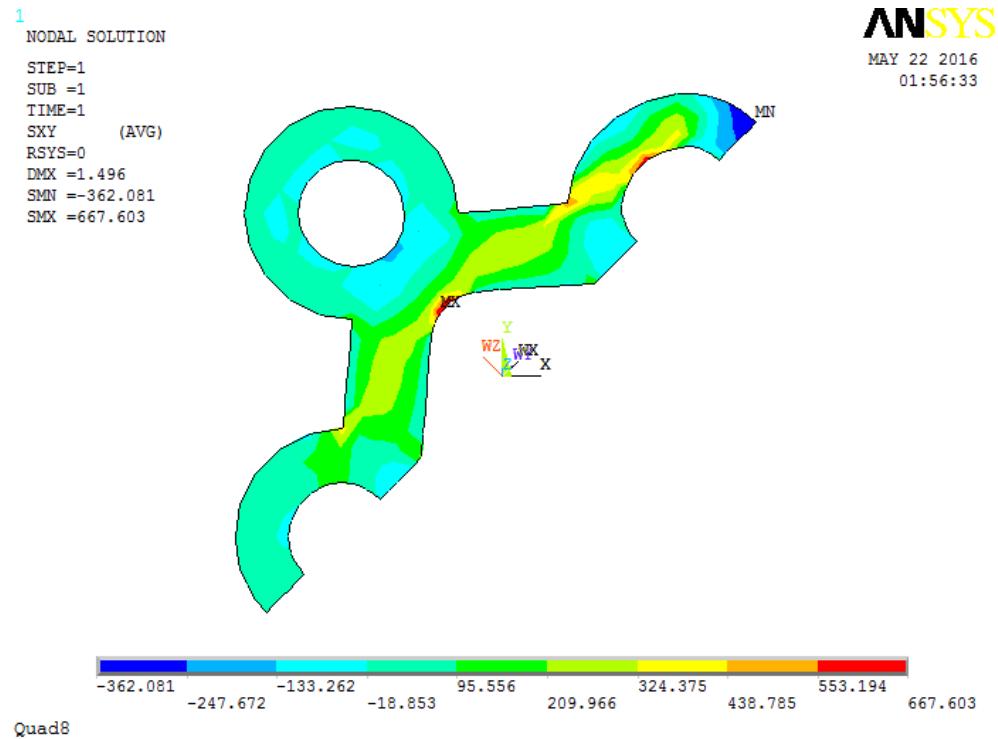


Figure 4.26: Mesh with Quad8. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

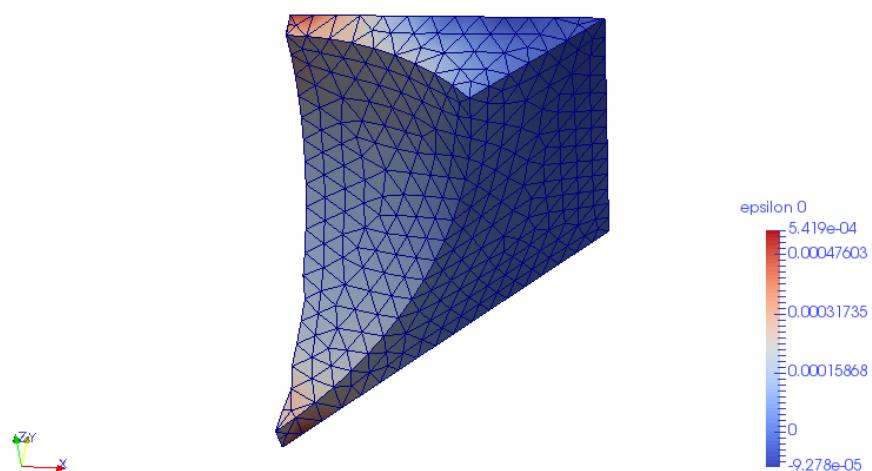
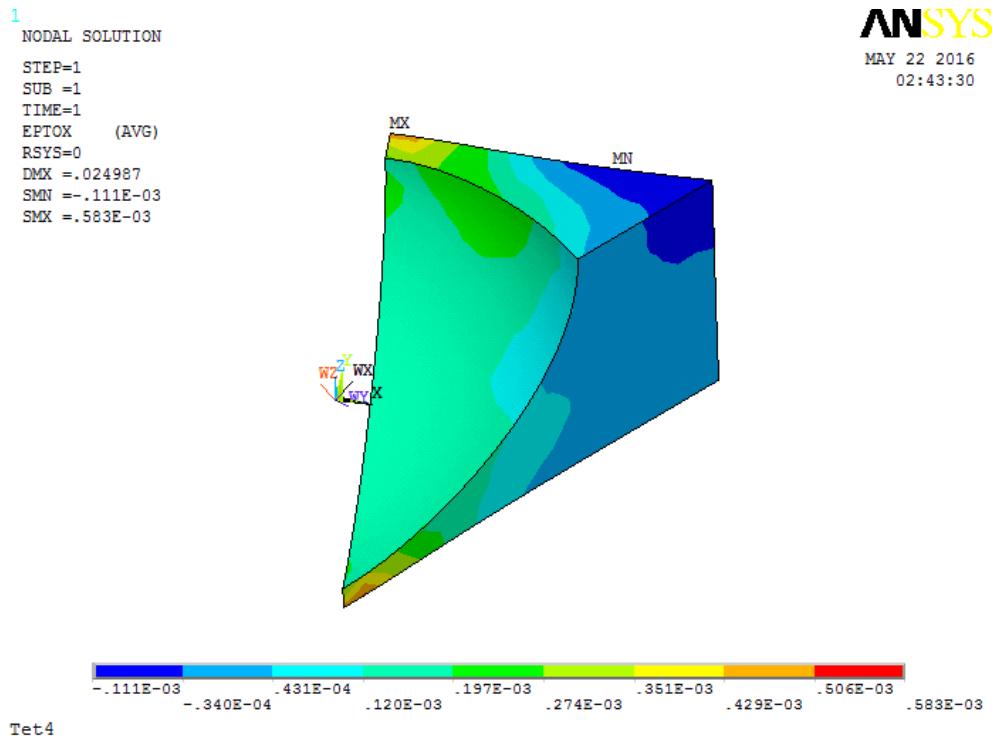


Figure 4.27: Mesh with Tet4. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

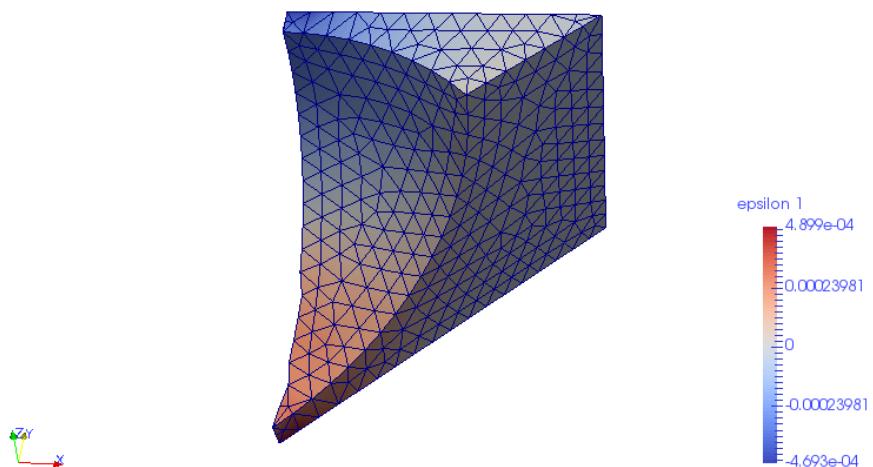
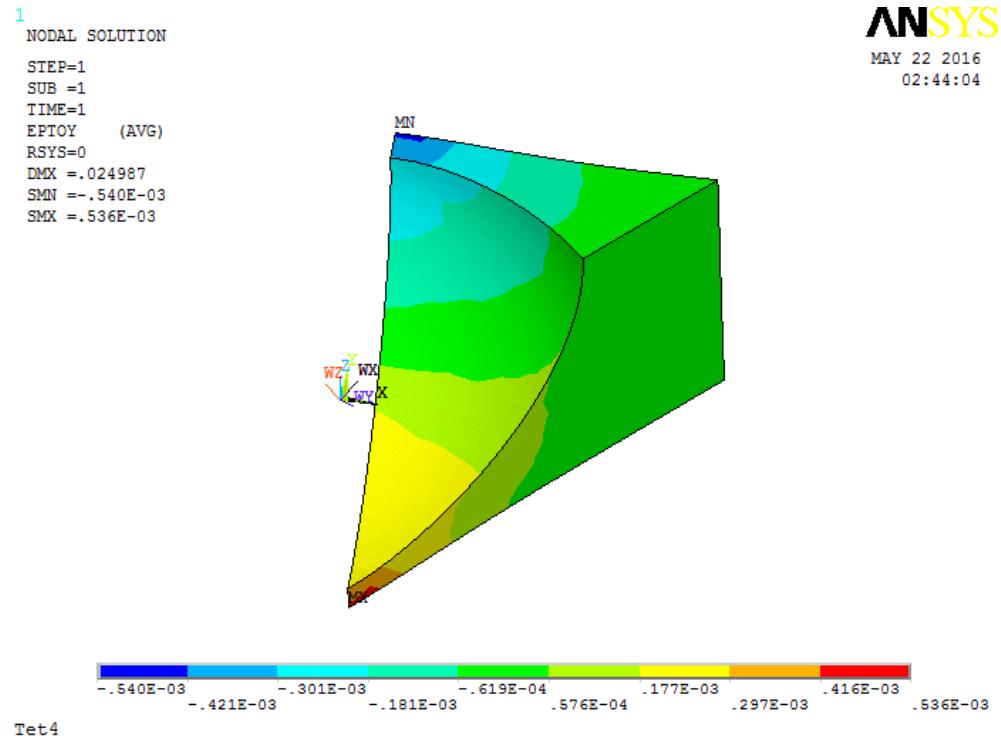


Figure 4.28: Mesh with Tet4. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

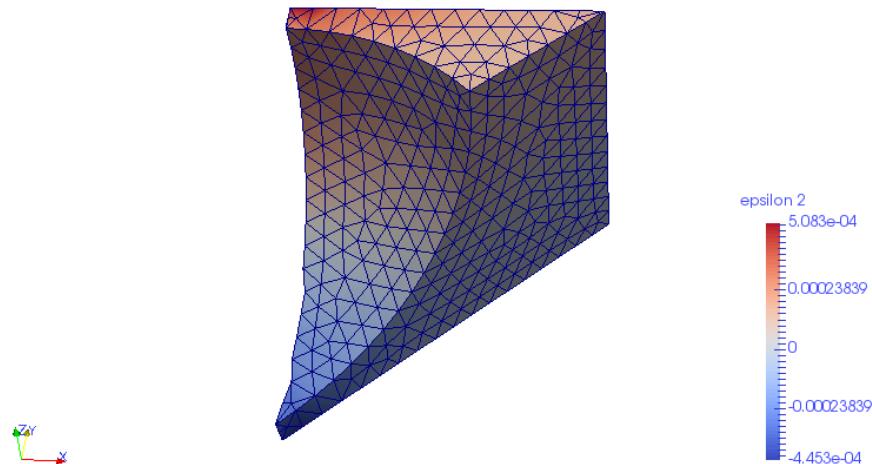
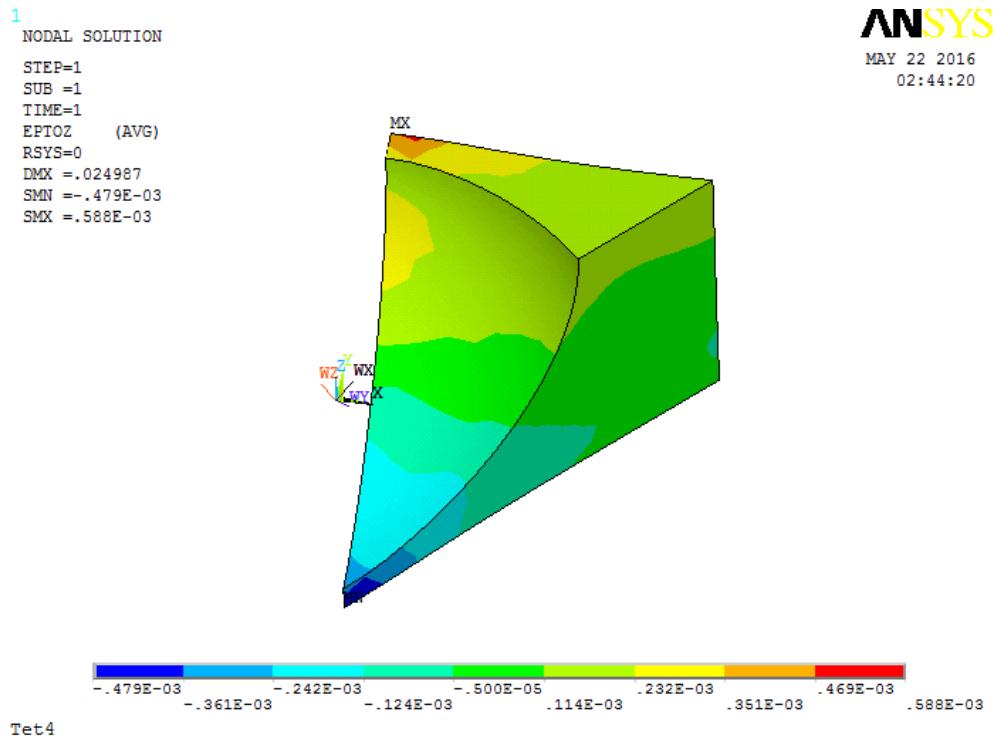


Figure 4.29: Mesh with Tet4. upper: contour plot of ϵ_{zz} from ANSYS; lower: contour plot of ϵ_{zz} calculated in AMfe, demonstrate in ParaView

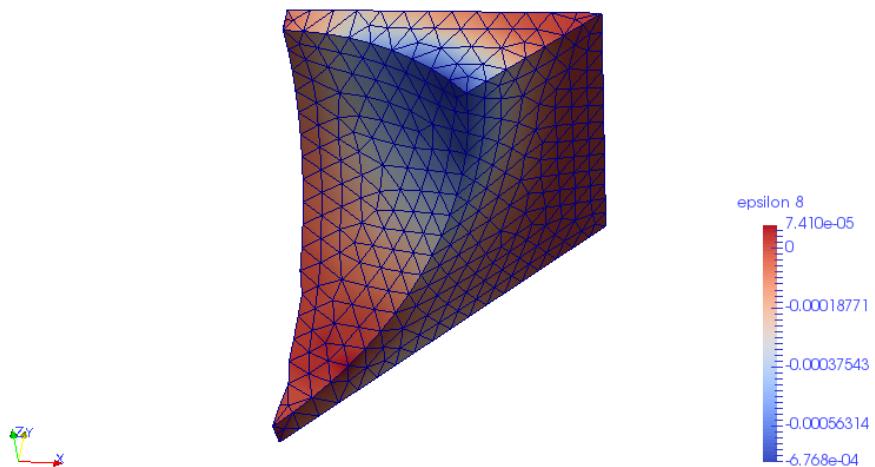
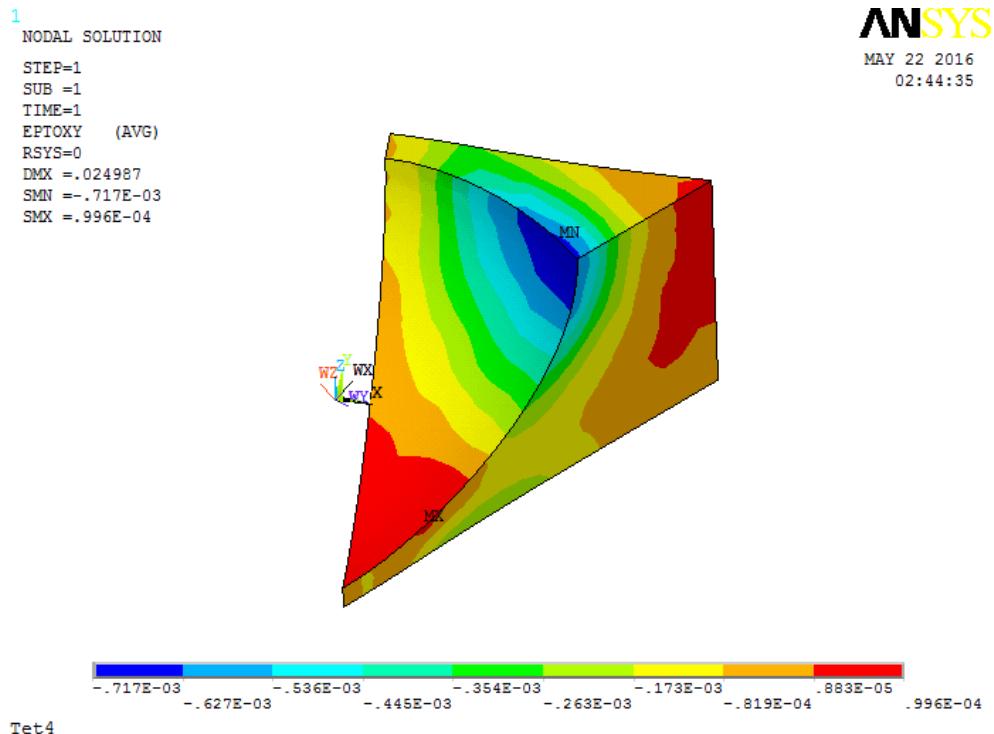


Figure 4.30: Mesh with Tet4. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

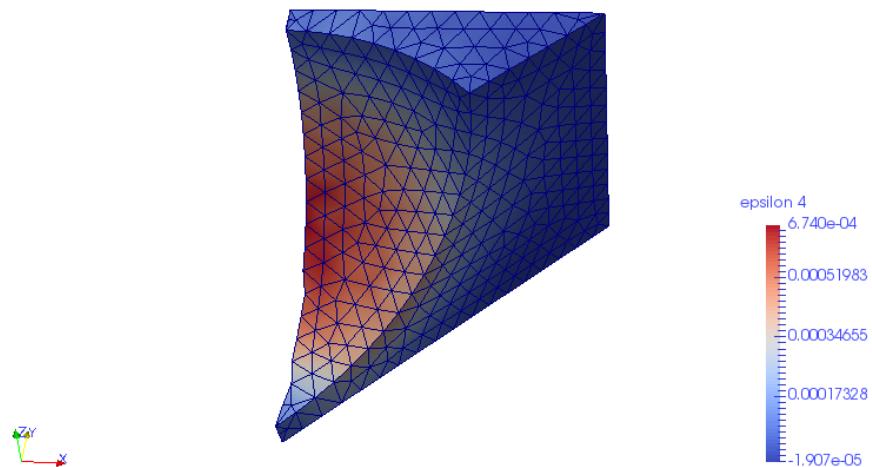
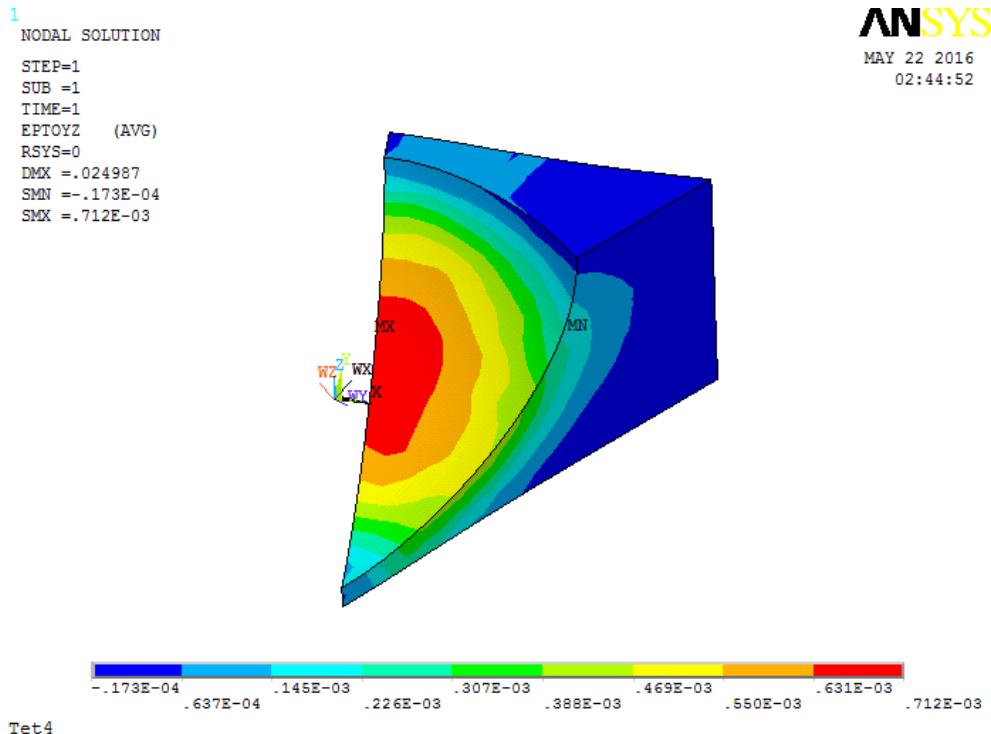


Figure 4.31: Mesh with Tet4. upper: contour plot of ϵ_{yz} from ANSYS; lower: contour plot of ϵ_{yz} calculated in AMfe, demonstrate in ParaView

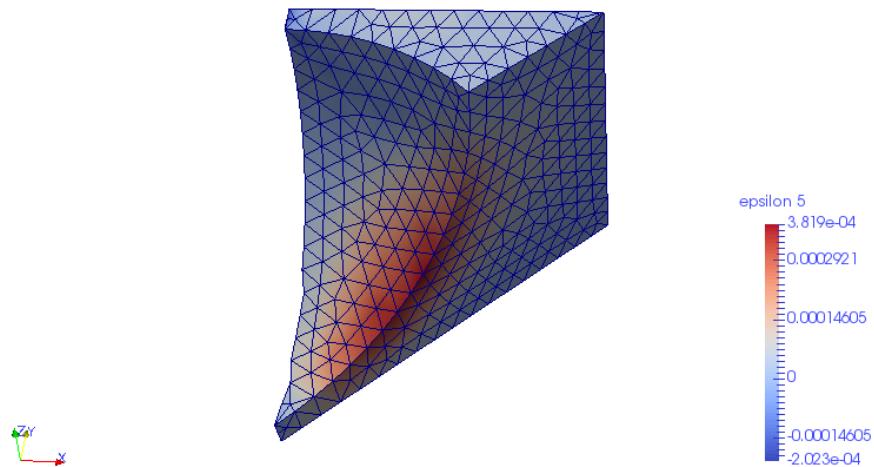
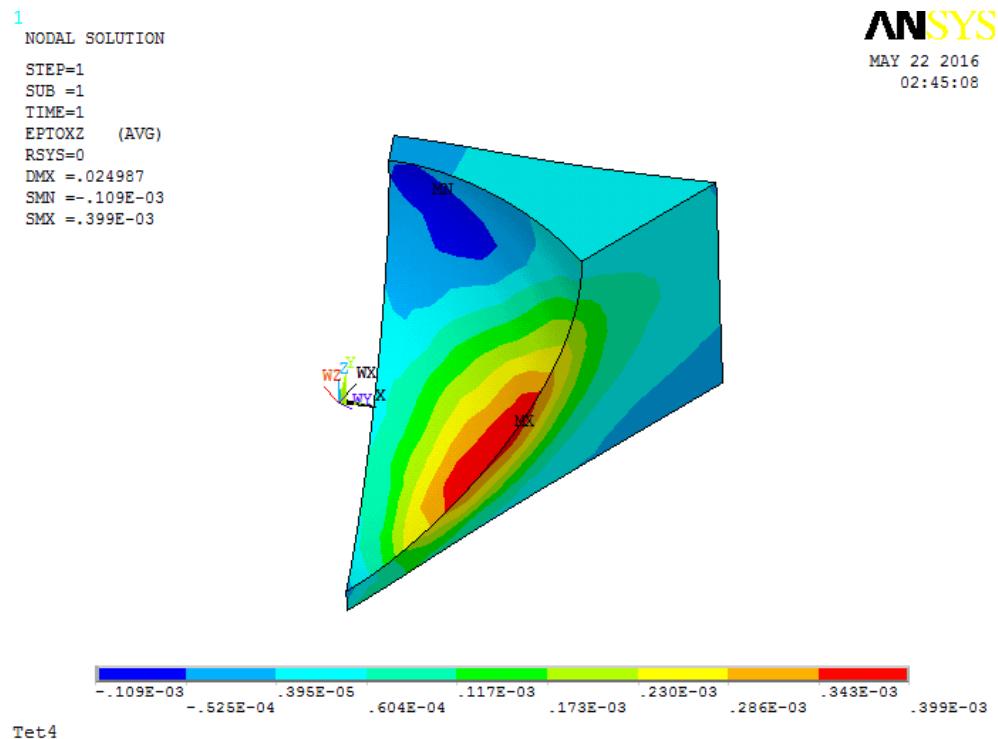


Figure 4.32: Mesh with Tet4. upper: contour plot of ϵ_{xz} from ANSYS; lower: contour plot of ϵ_{xz} calculated in AMfe, demonstrate in ParaView

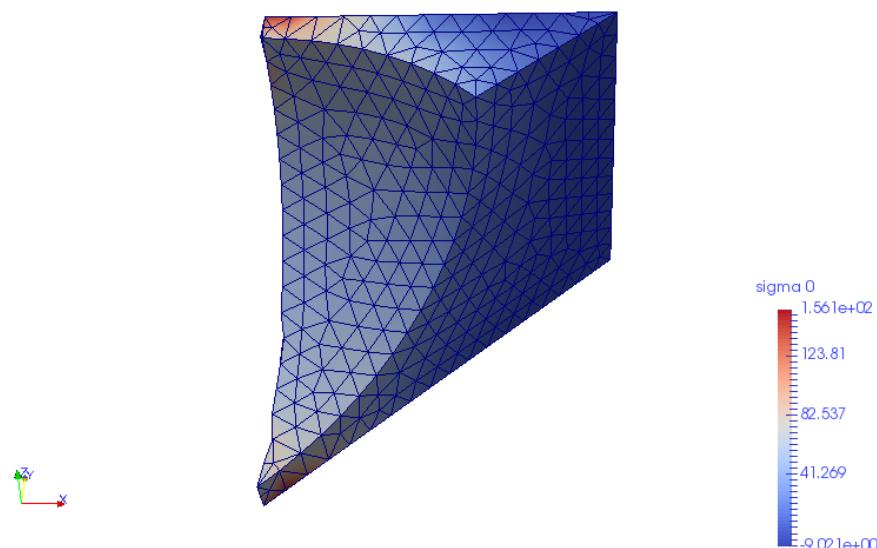
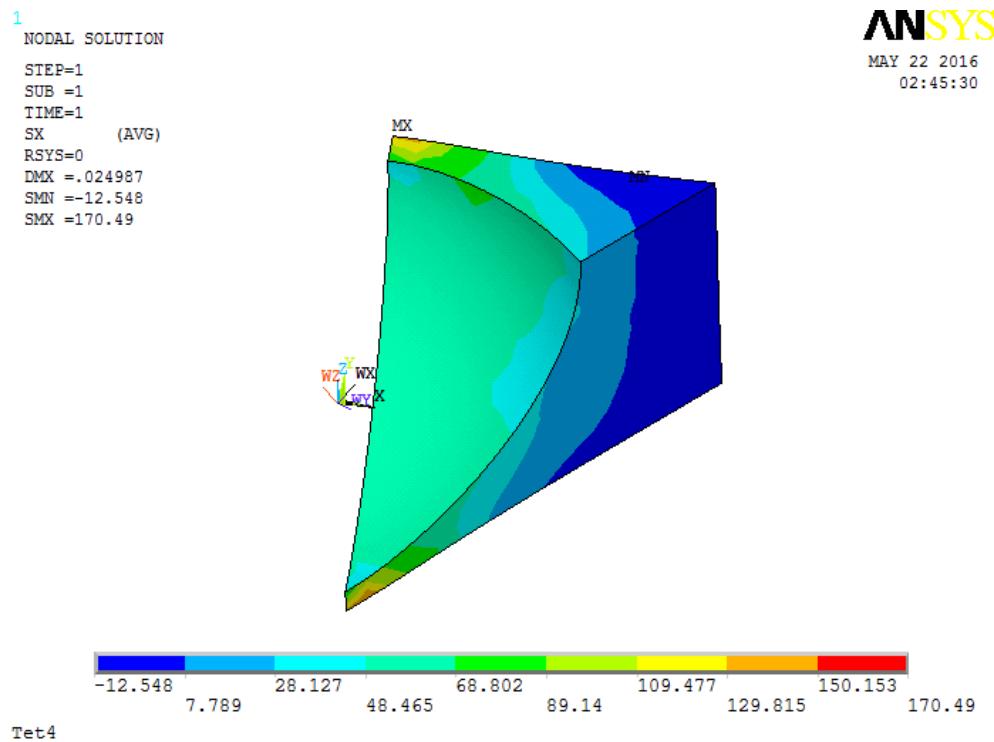


Figure 4.33: Mesh with Tet4. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView

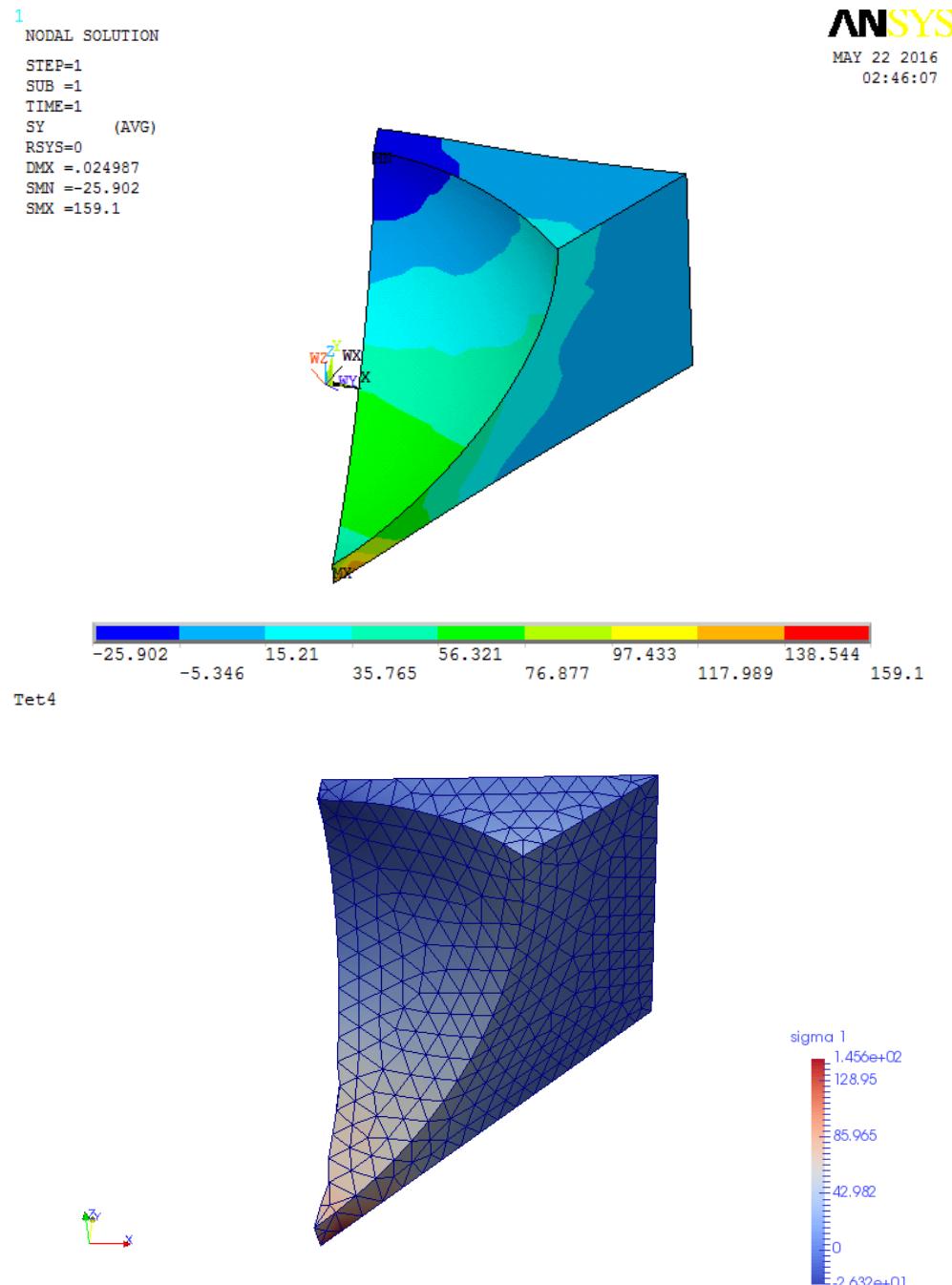


Figure 4.34: Mesh with Tet4. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

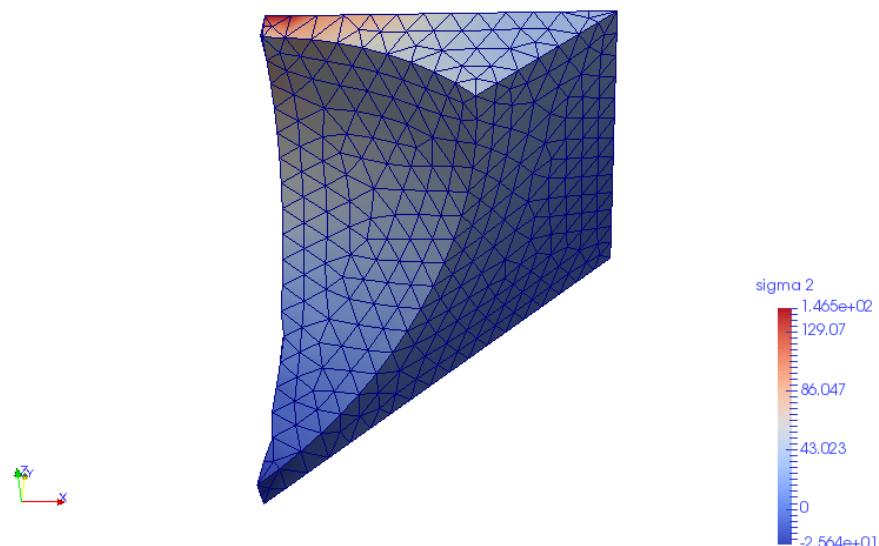
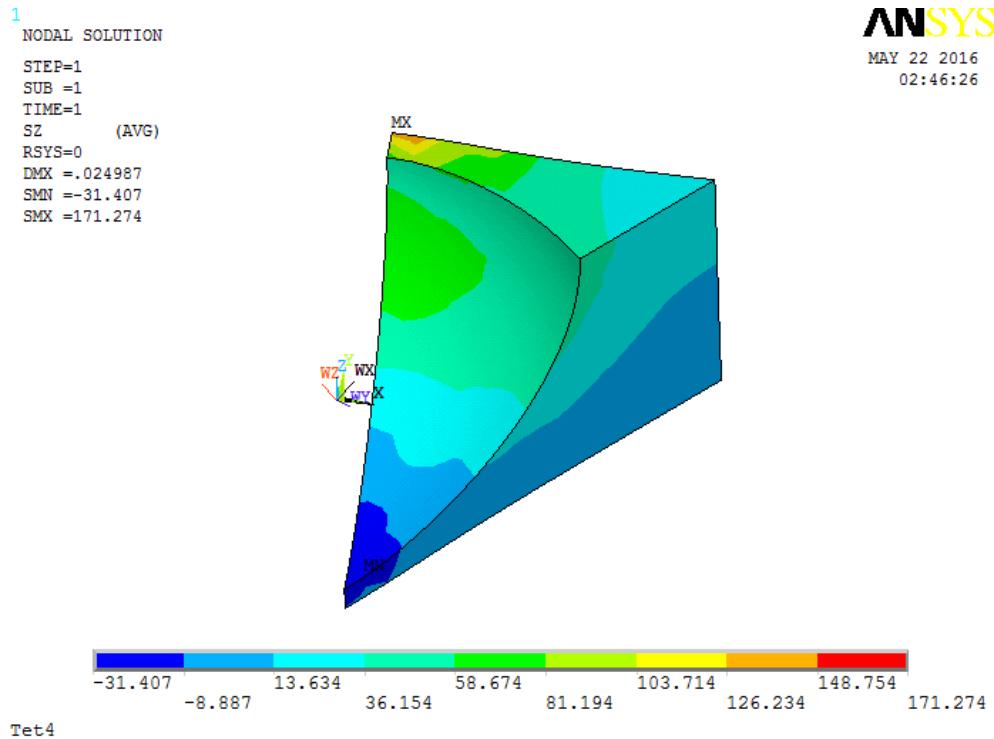


Figure 4.35: Mesh with Tet4. upper: contour plot of σ_{zz} from ANSYS; lower: contour plot of σ_{zz} calculated in AMfe, demonstrate in ParaView

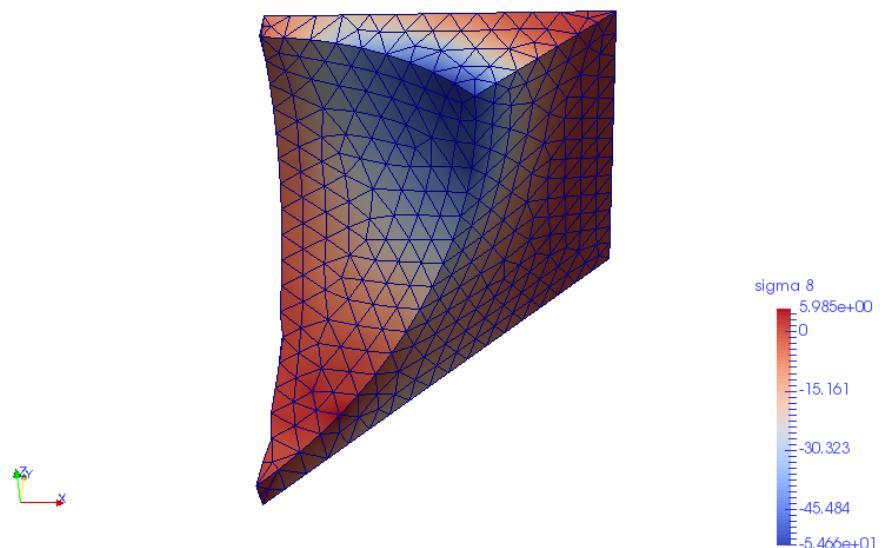
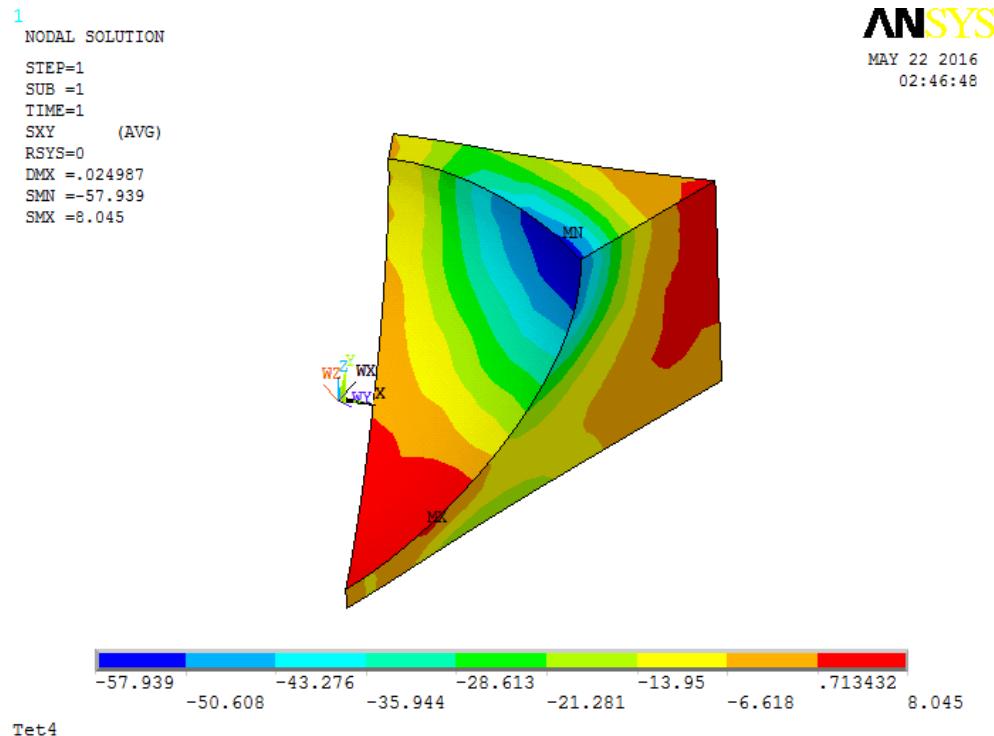


Figure 4.36: Mesh with Tet4. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

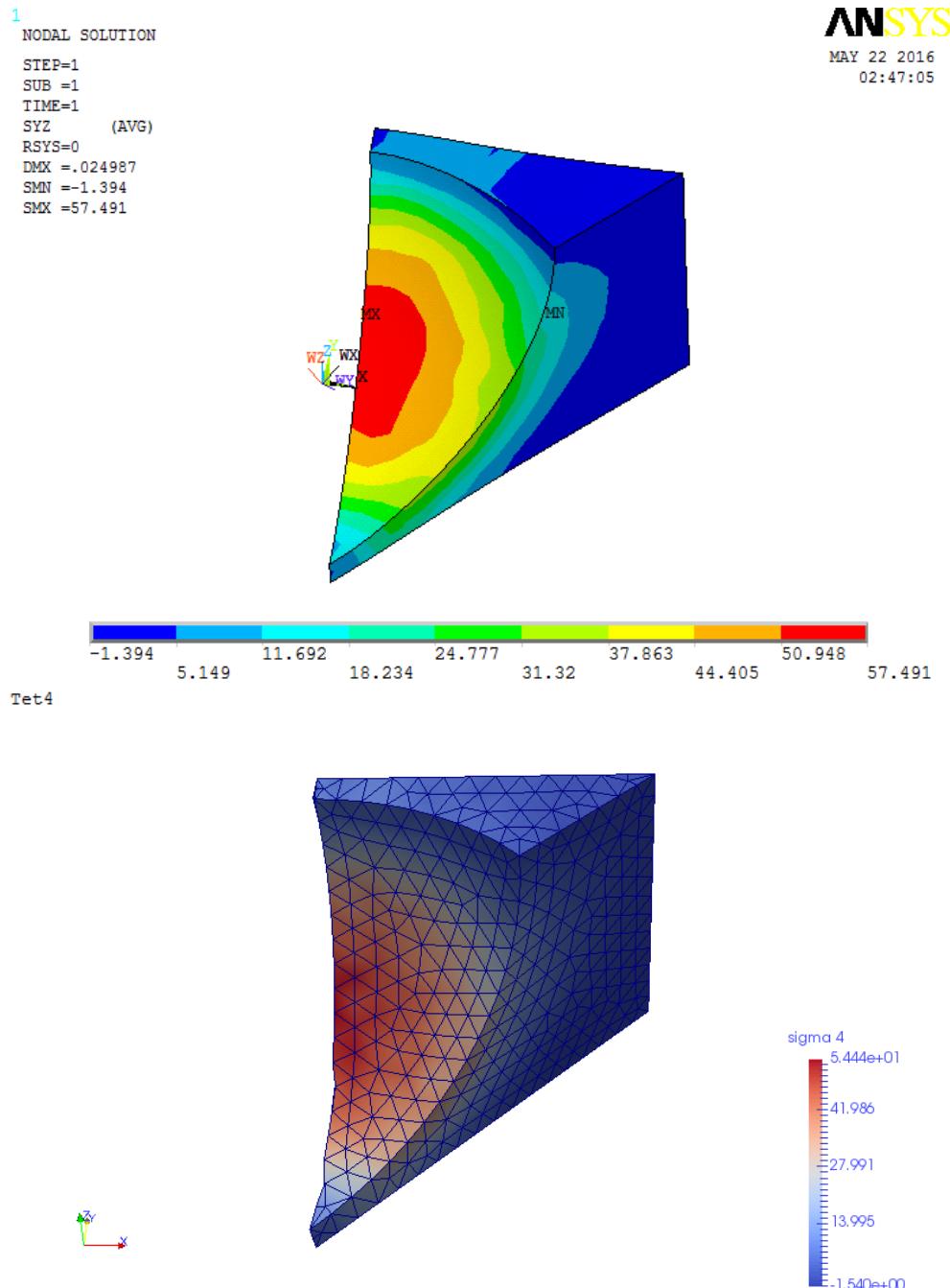


Figure 4.37: Mesh with Tet4. upper: contour plot of σ_{yz} from ANSYS; lower: contour plot of σ_{yz} calculated in AMfe, demonstrate in ParaView

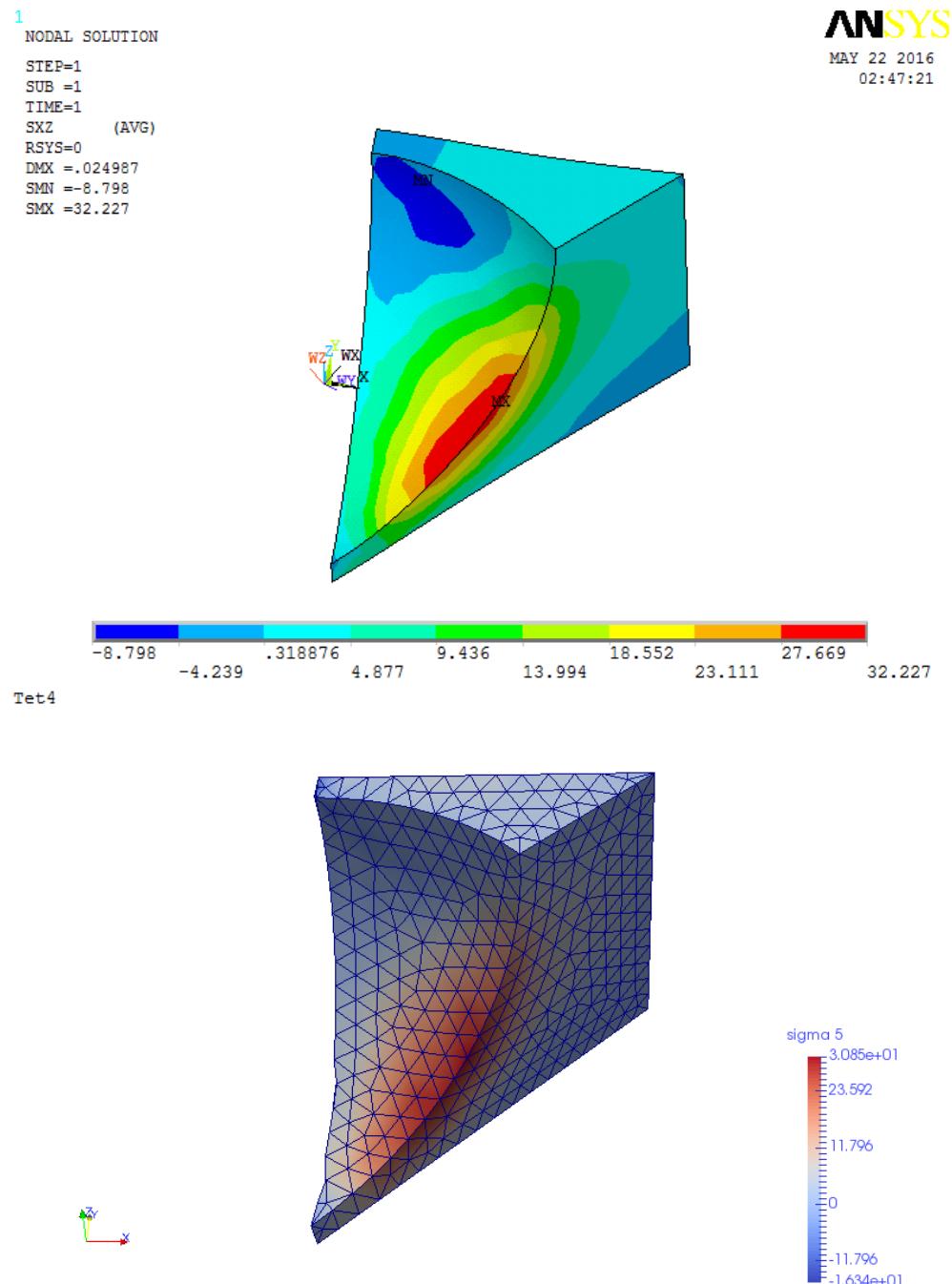


Figure 4.38: Mesh with Tet4. upper: contour plot of σ_{xz} from ANSYS; lower: contour plot of σ_{xz} calculated in AMfe, demonstrate in ParaView

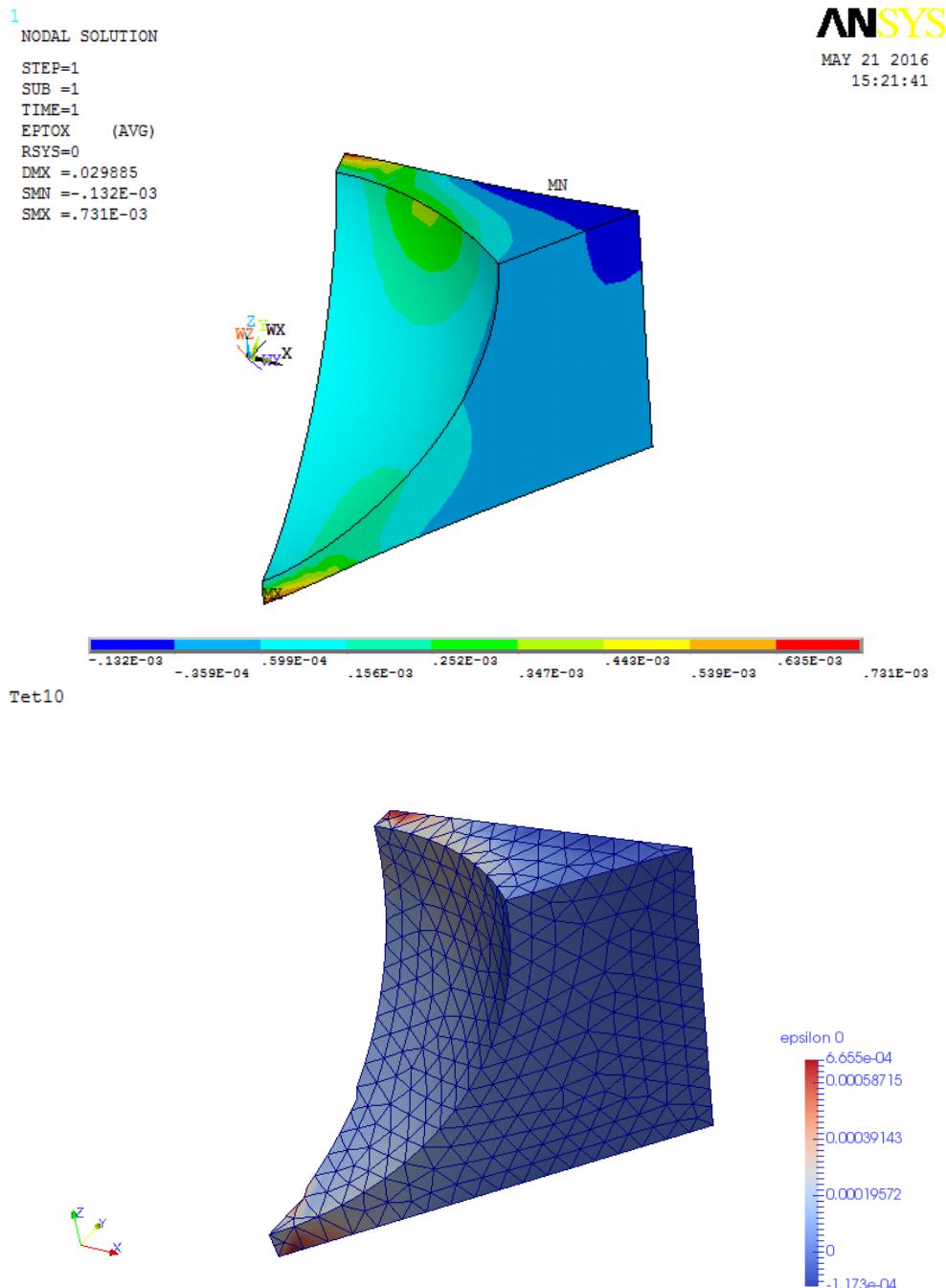


Figure 4.39: Mesh with Tet10. upper: contour plot of ϵ_{xx} from ANSYS; lower: contour plot of ϵ_{xx} calculated in AMfe, demonstrate in ParaView

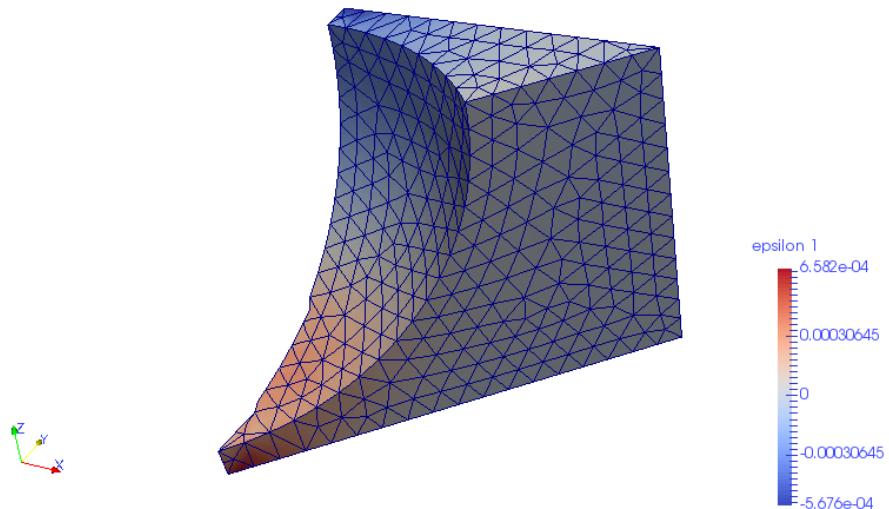
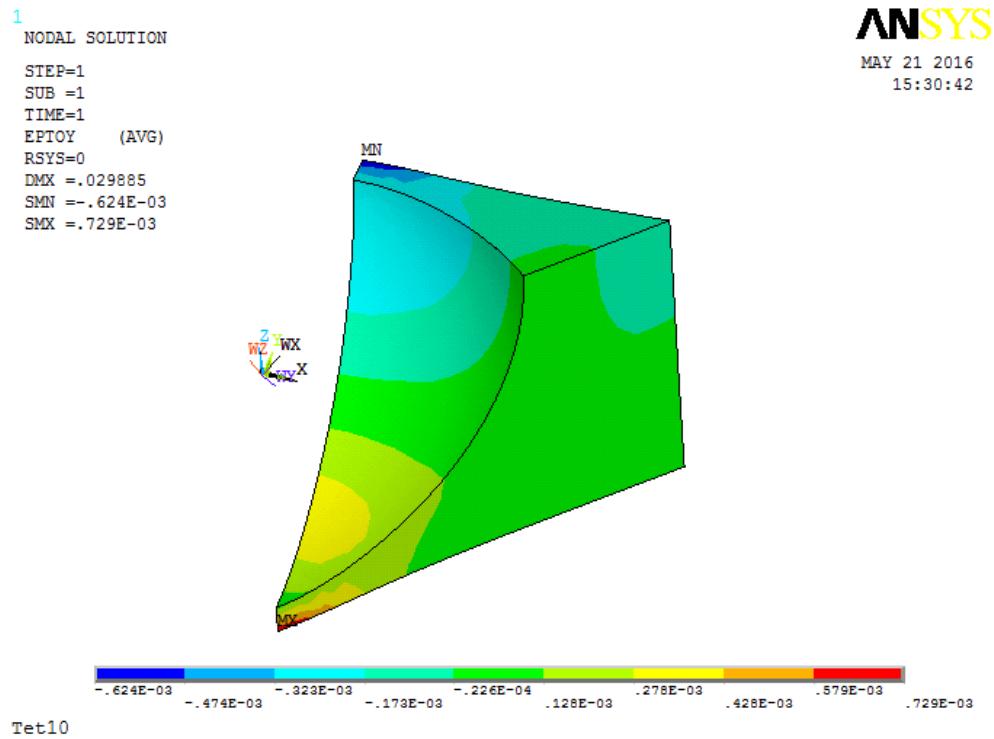


Figure 4.40: Mesh with Tet10. upper: contour plot of ϵ_{yy} from ANSYS; lower: contour plot of ϵ_{yy} calculated in AMfe, demonstrate in ParaView

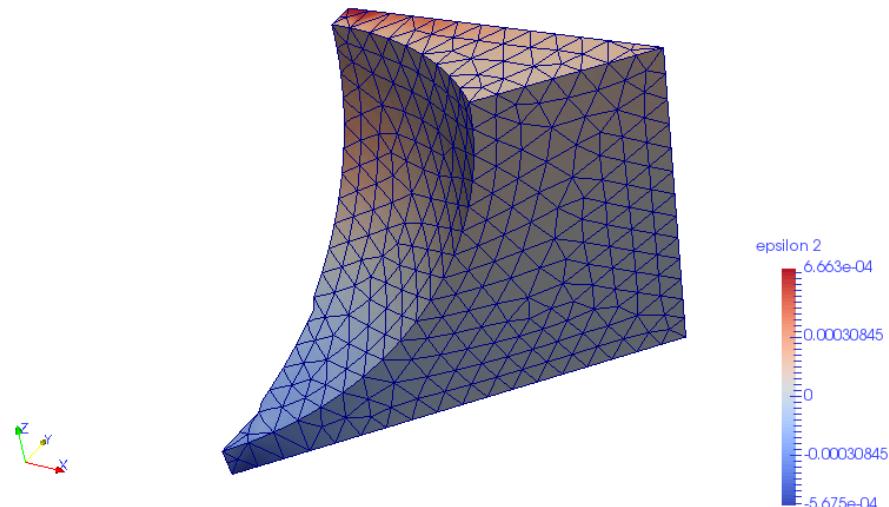
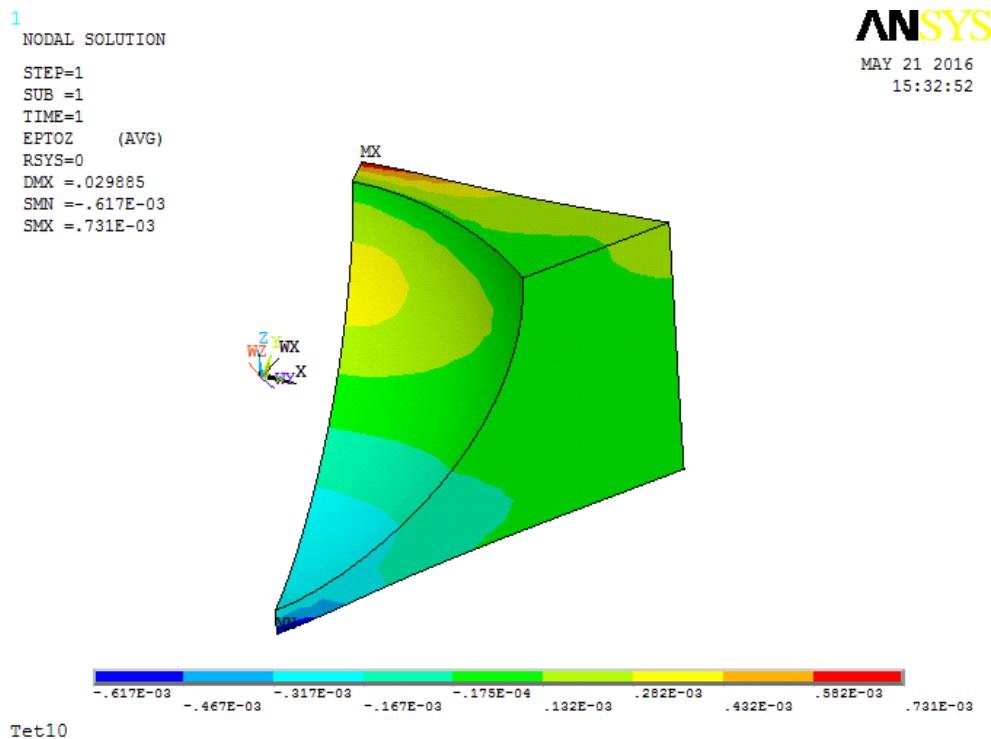


Figure 4.41: Mesh with Tet10. upper: contour plot of ϵ_{zz} from ANSYS; lower: contour plot of ϵ_{zz} calculated in AMfe, demonstrate in ParaView

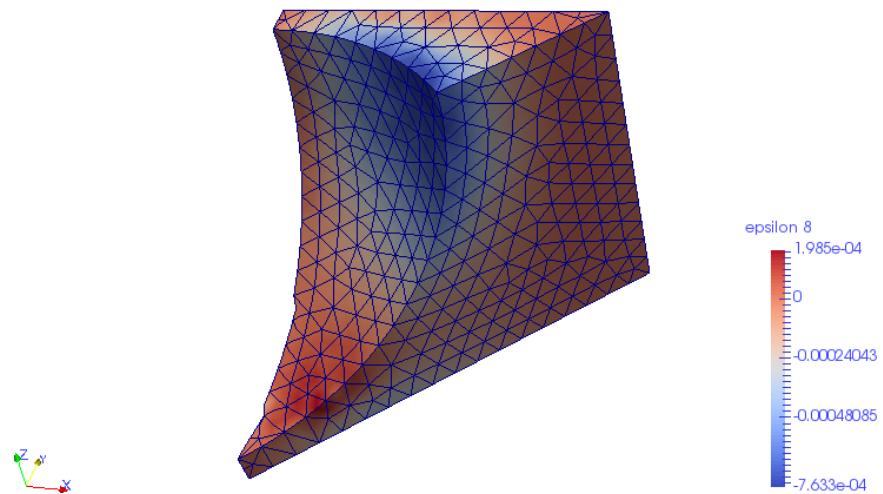
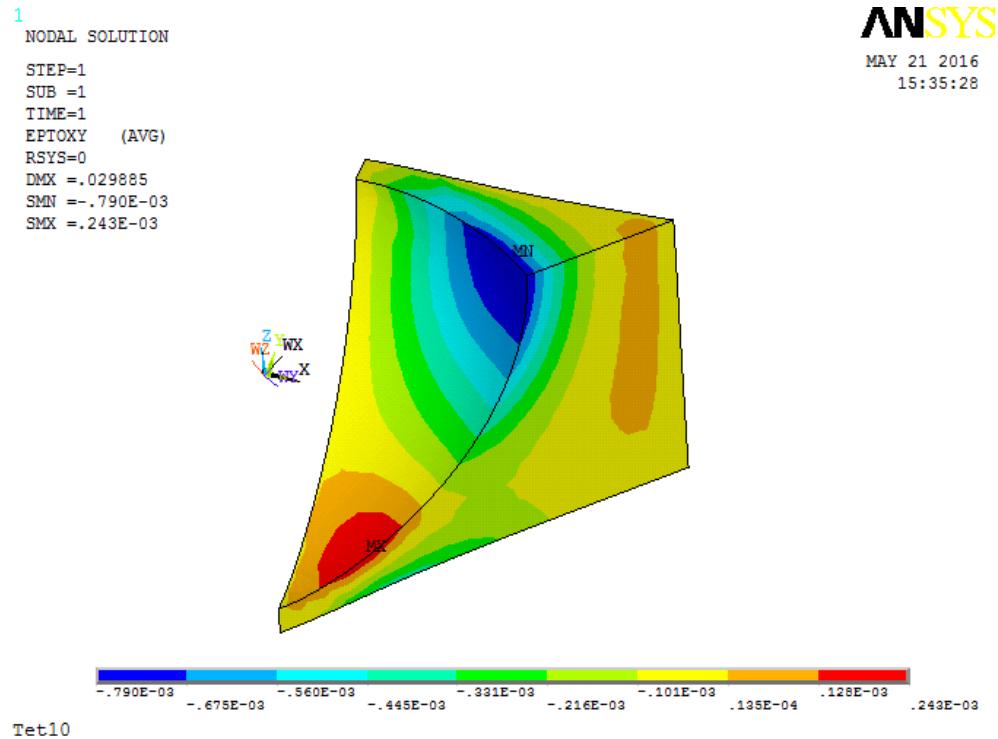


Figure 4.42: Mesh with Tet10. upper: contour plot of ϵ_{xy} from ANSYS; lower: contour plot of ϵ_{xy} calculated in AMfe, demonstrate in ParaView

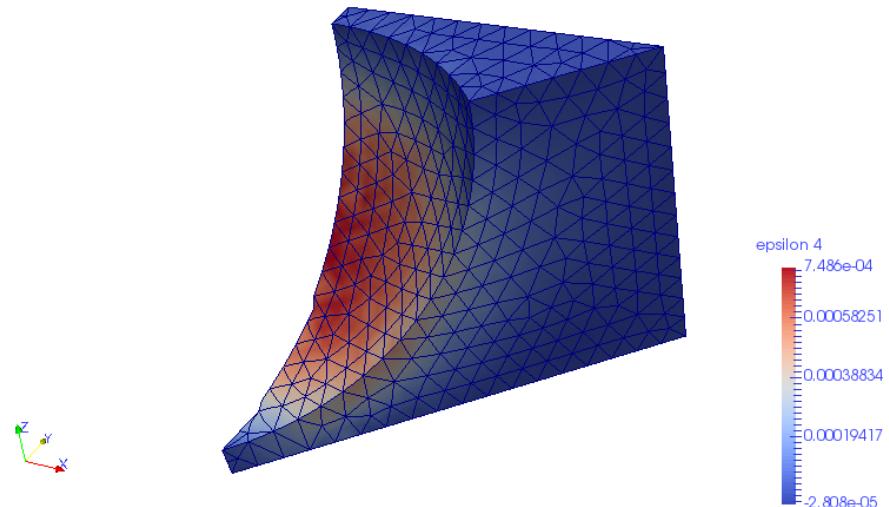
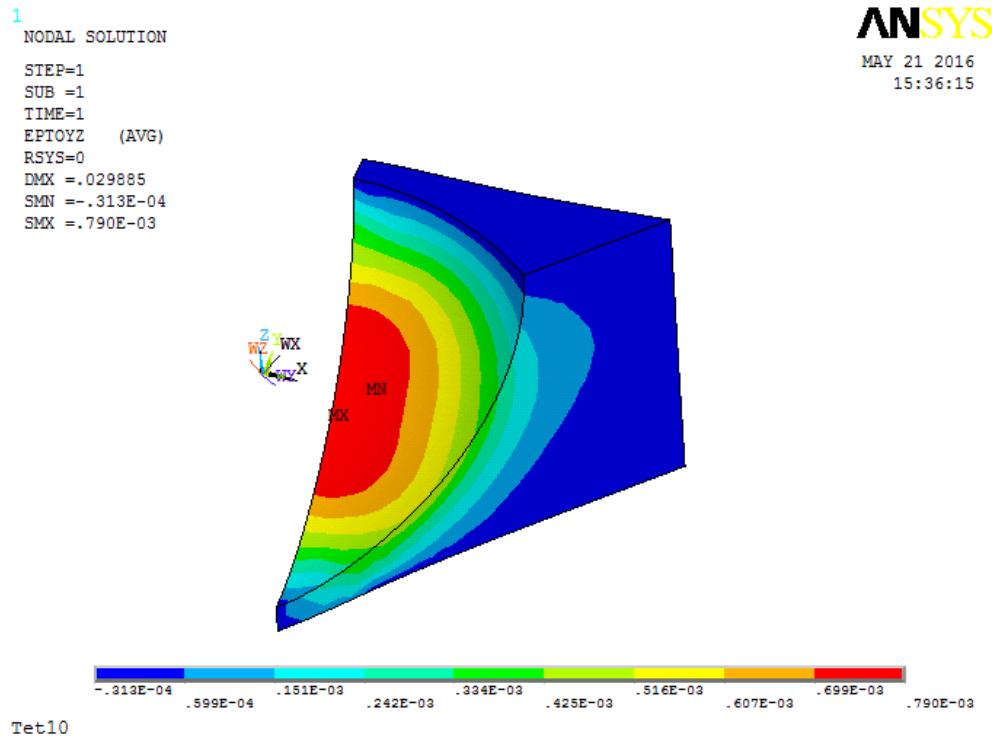


Figure 4.43: Mesh with Tet10. upper: contour plot of ϵ_{yz} from ANSYS; lower: contour plot of ϵ_{yz} calculated in AMfe, demonstrate in ParaView

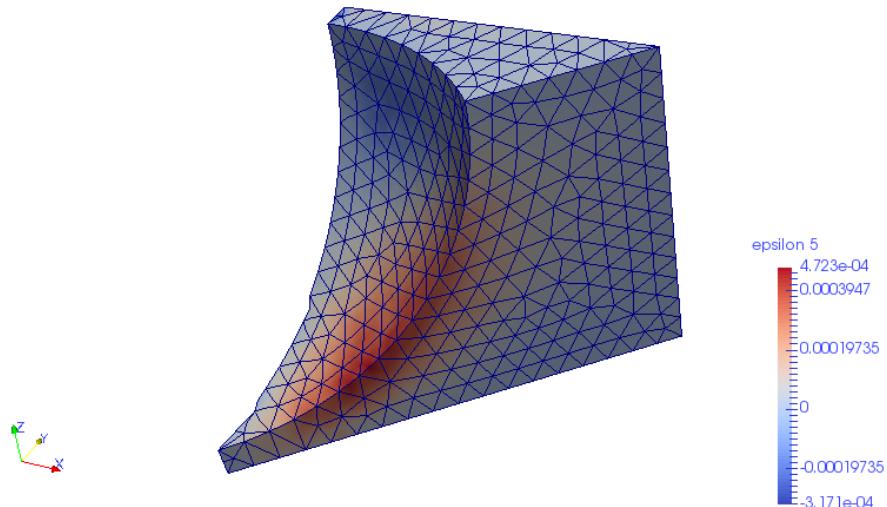
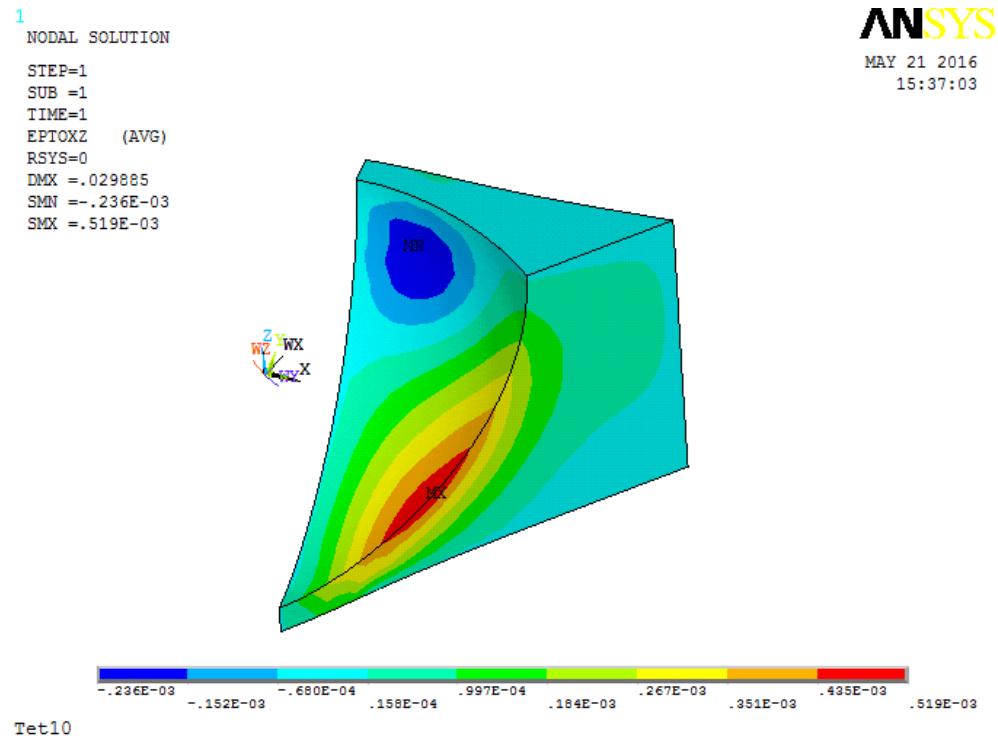


Figure 4.44: Mesh with Tet10. upper: contour plot of ϵ_{xz} from ANSYS; lower: contour plot of ϵ_{xz} calculated in AMfe, demonstrate in ParaView

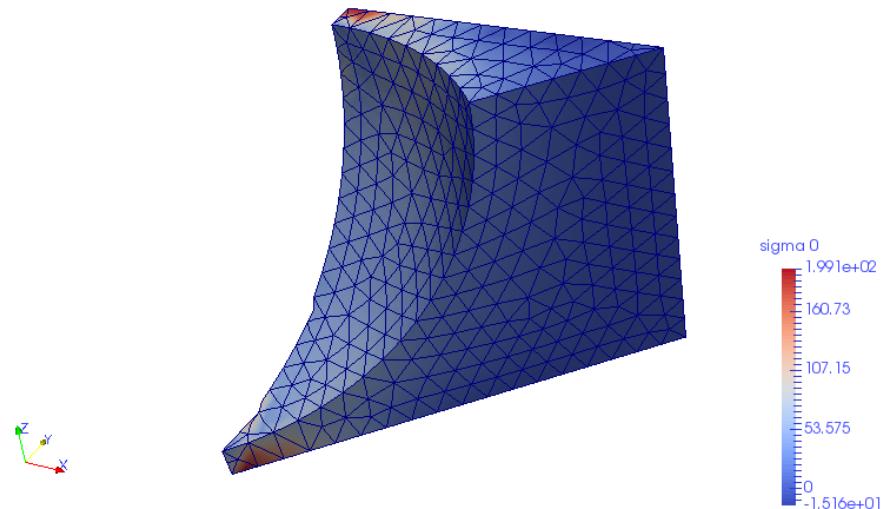
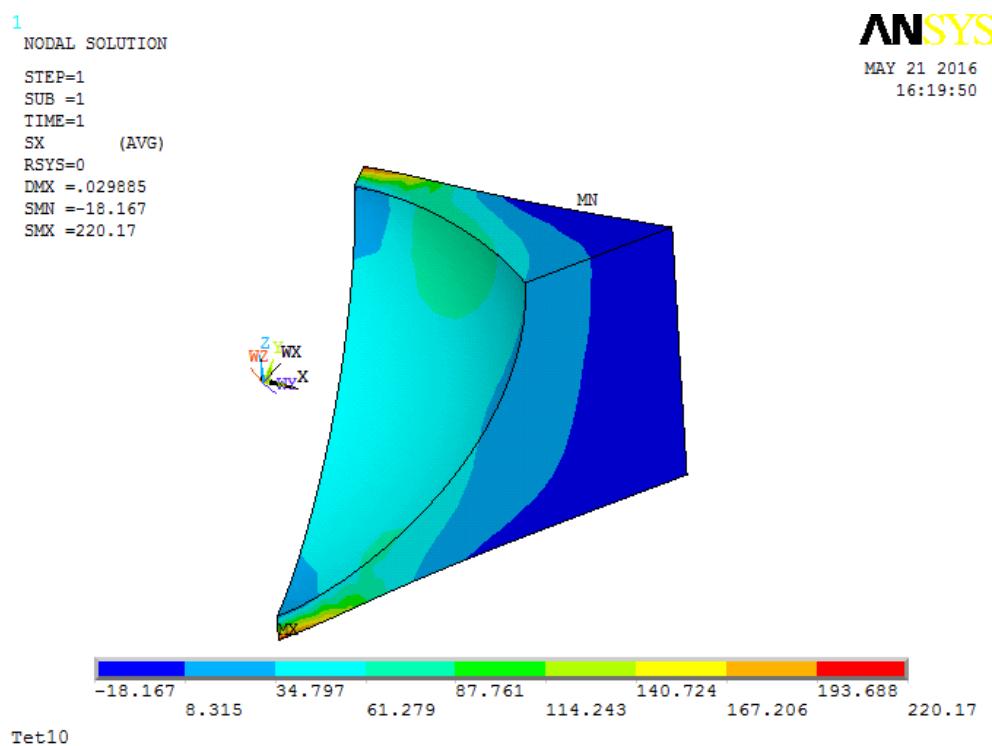


Figure 4.45: Mesh with Tet10. upper: contour plot of σ_{xx} from ANSYS; lower: contour plot of σ_{xx} calculated in AMfe, demonstrate in ParaView

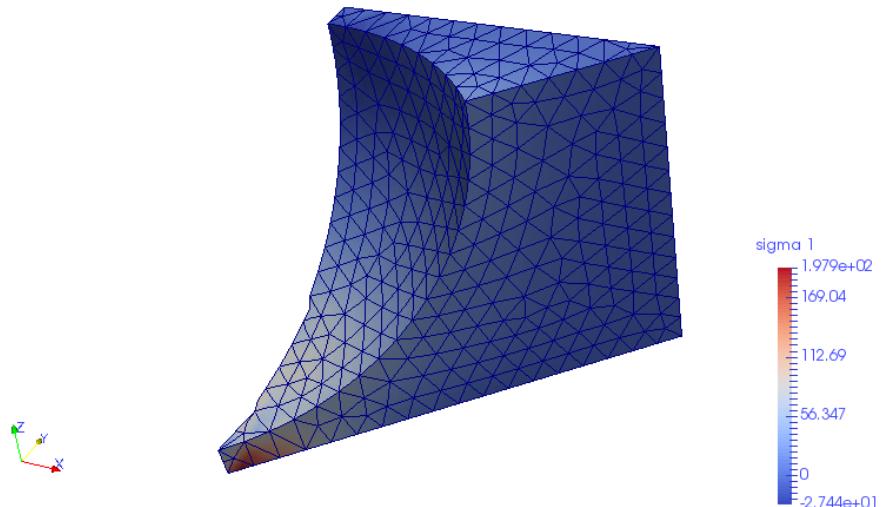
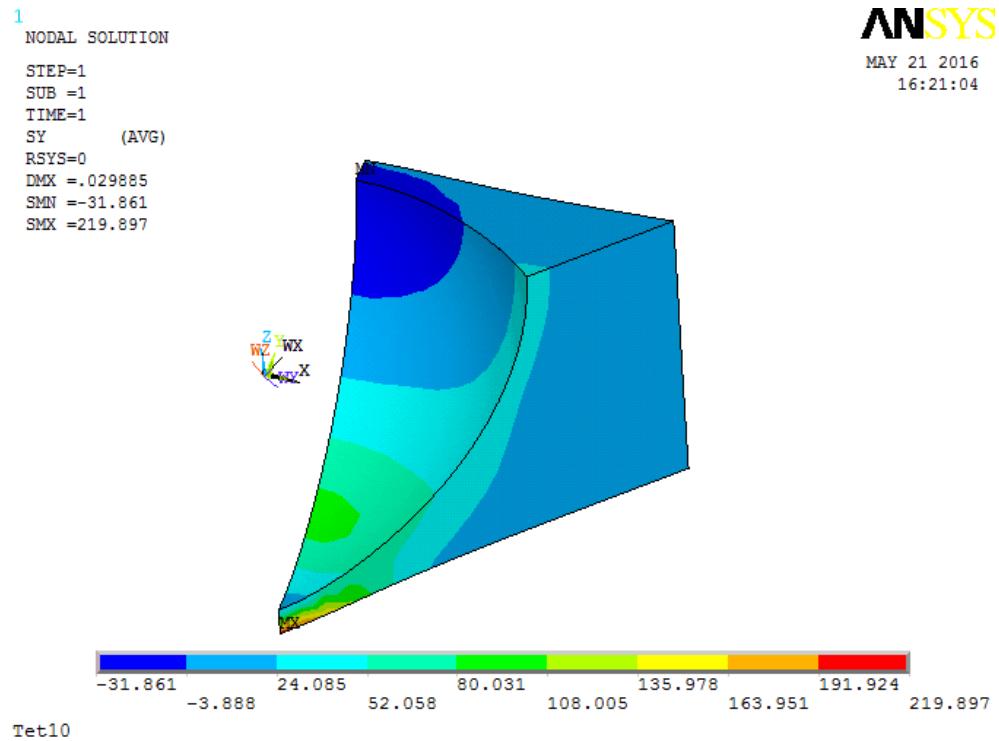


Figure 4.46: Mesh with Tet10. upper: contour plot of σ_{yy} from ANSYS; lower: contour plot of σ_{yy} calculated in AMfe, demonstrate in ParaView

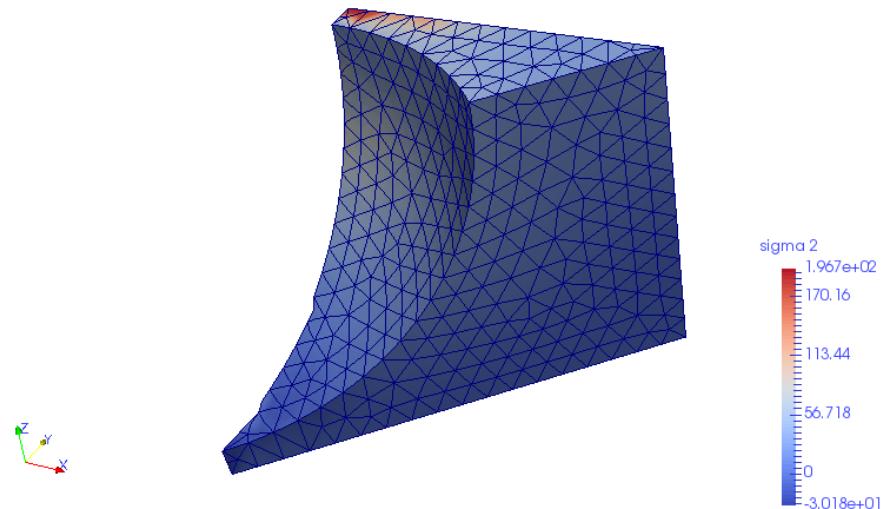
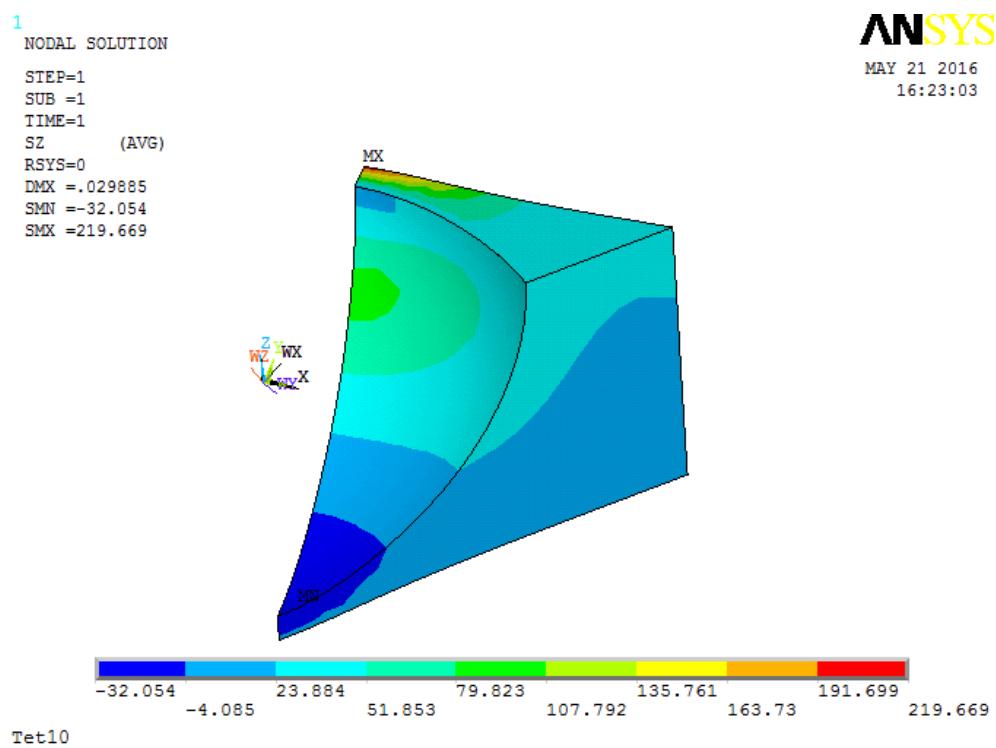


Figure 4.47: Mesh with Tet10. upper: contour plot of σ_{zz} from ANSYS; lower: contour plot of σ_{zz} calculated in AMfe, demonstrate in ParaView

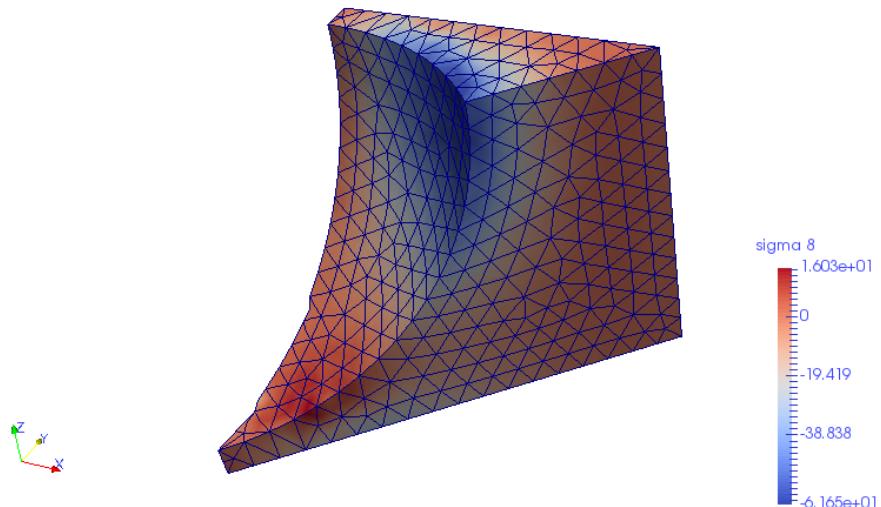
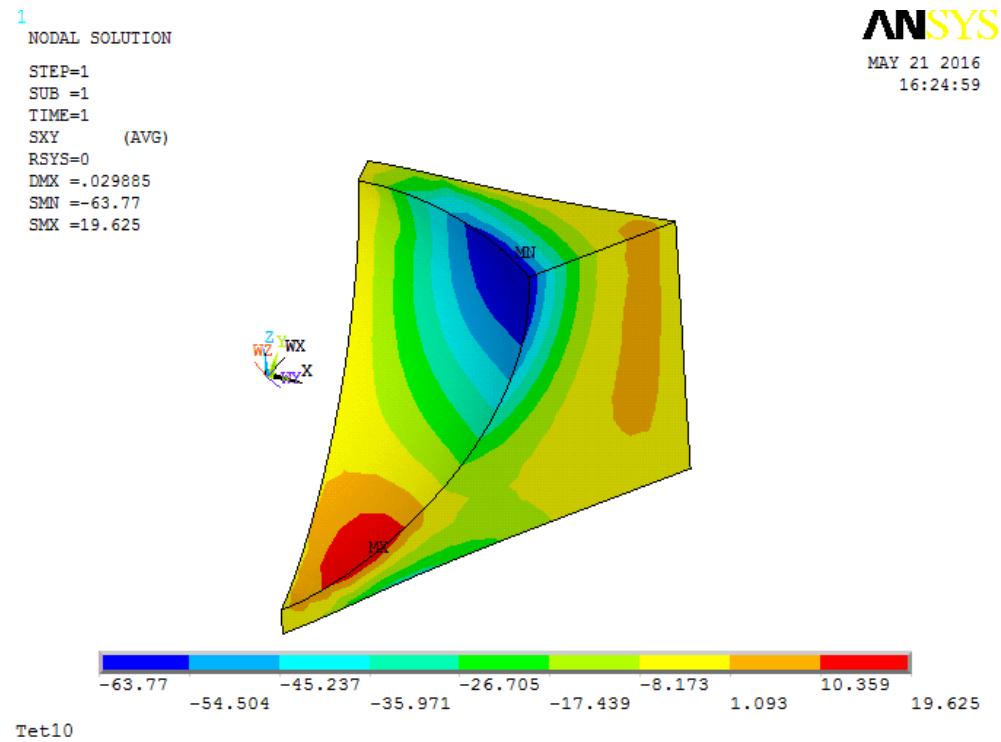


Figure 4.48: Mesh with Tet10. upper: contour plot of σ_{xy} from ANSYS; lower: contour plot of σ_{xy} calculated in AMfe, demonstrate in ParaView

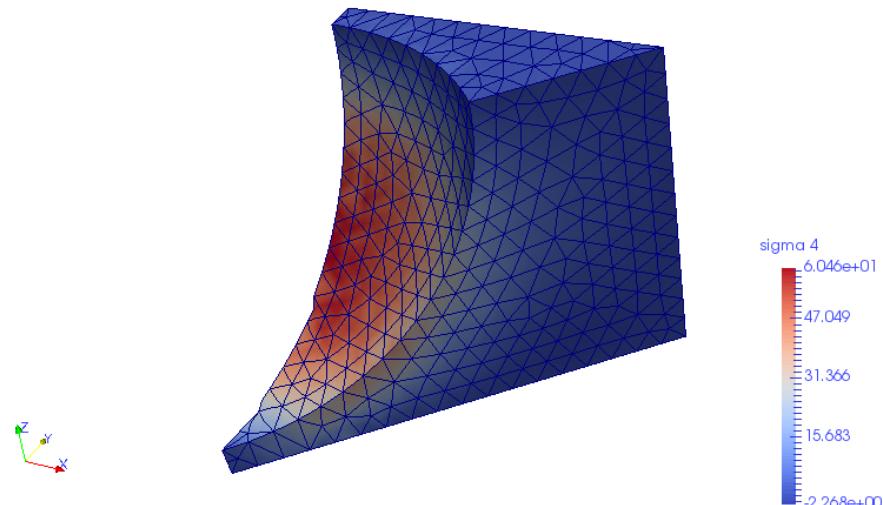
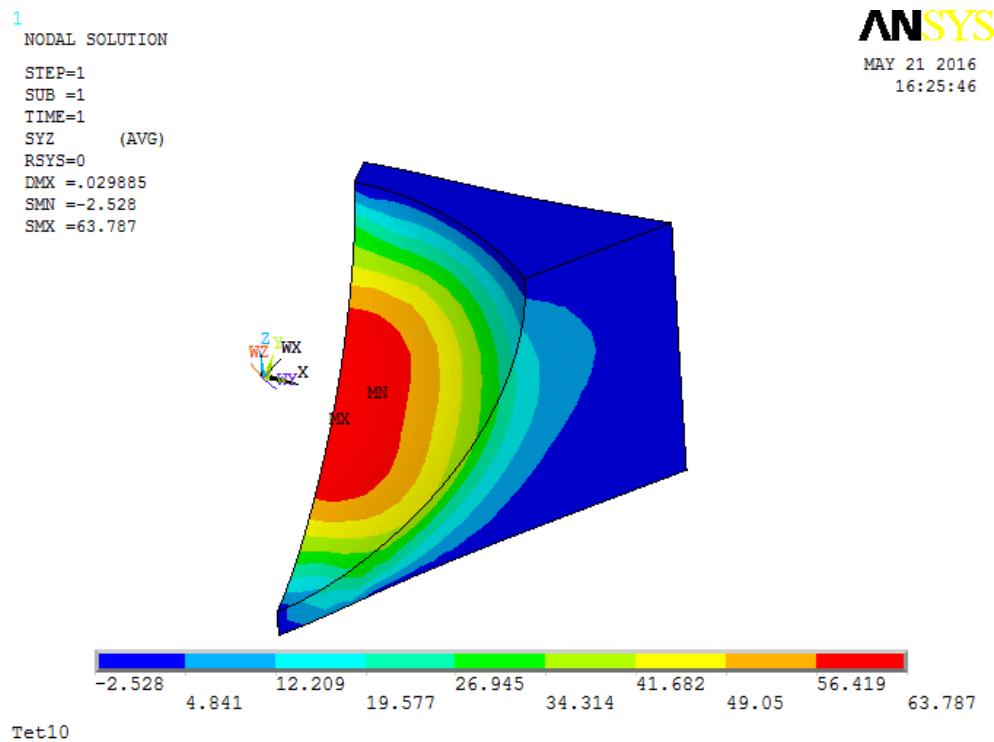


Figure 4.49: Mesh with Tet10. upper: contour plot of σ_{yz} from ANSYS; lower: contour plot of σ_{yz} calculated in AMfe, demonstrate in ParaView

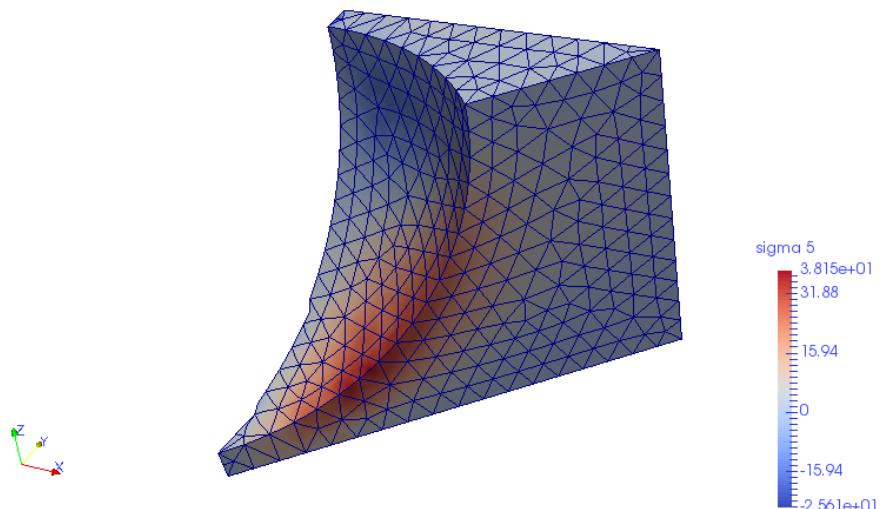
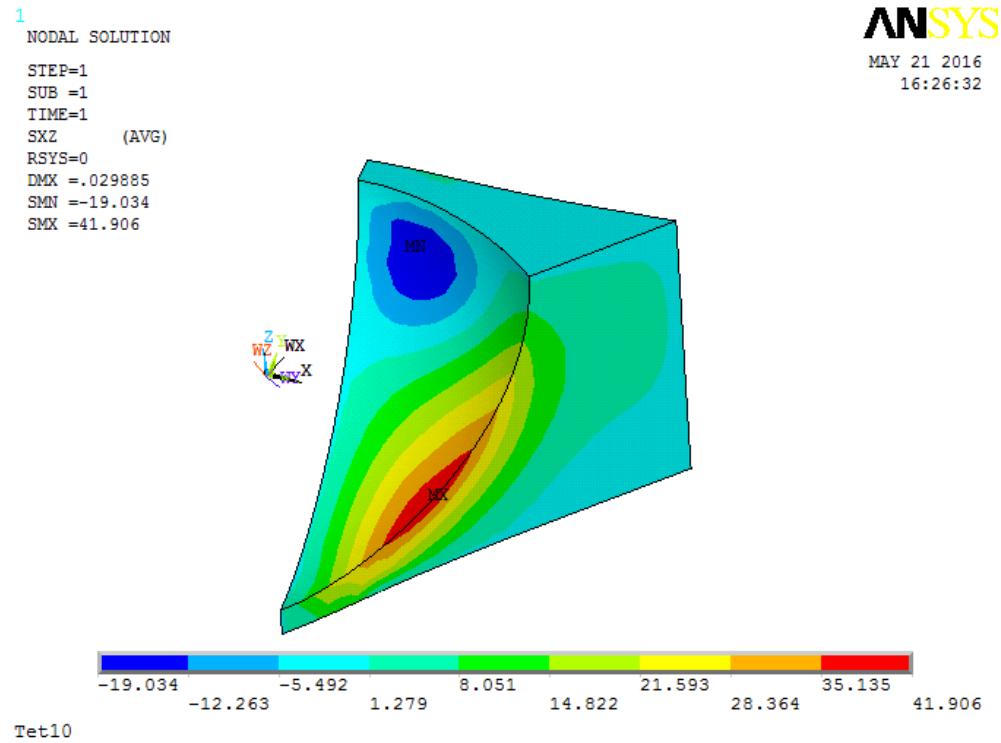


Figure 4.50: Mesh with Tet10. upper: contour plot of σ_{xz} from ANSYS; lower: contour plot of σ_{xz} calculated in AMfe, demonstrate in ParaView

4.4 Convergence Analysis

We can either decrease the element size or increase the polynomial degree of the shape function to approach an exact solution. This process is denoted as convergence. A convergence analysis provides a function to reduce the relative error of the solution with increasing number of degree of freedom [Wall 2014]. An example of convergence analysis is illustrated in Figure 4.51. Because of the technical restriction, we can only export the node, element and displacement lists from ANSYS and set all of them as input to AMfe Toolbox for solving strain and stress. We can not do the process conversely to solve the displacement by using prescribed Neumann boundary condition and Dirichlet boundary condition as input. It is still a feasible plan for future research. On the other hand, we can analyse this problem in a global point of view. It is also necessary to check the convergence of internal energy. The internal energy can be derived as:

$$\prod int = \frac{1}{2} \int_{\Omega} \sigma_{ij}^u \epsilon_{ij}^u d\Omega \quad (4.3)$$

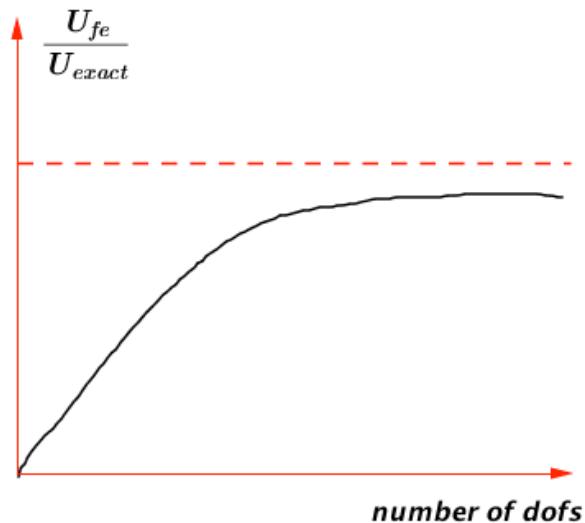


Figure 4.51: Convergence analysis

bibid Interelement Averaging can be as future research .. And I can talk about why i choose extrapolation, why not direct evaluation! Rutzmoser 2016, p. 1-10

Bibliography

- Dukkipati, Rao V. (2010). *Numerical Methods*. 4835/24 Ansari Road, Daryaganj, New Delhi: New Age International Publishers.
- Wall, W. A. (2014). *Finite Element*. Ed. by Svenja Schoeder. 9th ed. Lehrstuhl für Numerische Mechanik, Fakultät für Maschinenwesen, Technische Universität München.
- 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)