NATIONAL TECHINCAL UNIVERSITY OF ATHENS

SCHOOL OF CIVIL ENGINEERING

ADERS Program

**MASTER THESIS**

# PATHOLOGICAL ISSUES OF The FINITE ELEMENT METHOD

**Ahmed Shokry**

BSc. Cairo University

**Supervisor**: Prof. K. Spiliopoulos

Athens, July 2014

NATIONAL TECHINCAL UNIVERSITY

OF ATHENS

SCHOOL OF CIVIL ENGINEERING

ADERS Program

# PATHOLOGICAL ISSUES OF THE FINITE ELEMENT METHOD

**MASTER THESIS**

Analysis and Design of Earthquake Resistant Structures

Submitted on 30.07.2014

From

Ahmed Mohamed Osama Shokry

BSc., Cairo University

11021253

Supervisor: Prof. K. Spiliopoulos

# Acknowledgment

In presenting this master thesis, I would like to express my gratitude to the following:

- My parents and sister for their continuous support.
- Prof. Spiliopoulos (*Statics and Aseismic Research Laboratory, National Technical University of Athens*) for suggesting and supervising my research.
- Prof. Papadrakakis (*Statics and Aseismic Research Laboratory & Coordinator of ADERS program, National Technical University of Athens*) for the successful coordination of ADERS program.
- Dr. Plivers (*Department of Structural Engineering, National Technical University of Athens*) for his continuous support during my life in Greece.
- Prof. Nayer El Essenawy, Dr. Sameh Mehanny and Dr. M. Hassanein Sirror (*Department of Structural Engineering, Cairo University*) for their beneficial advices and continuous support during my undergraduate and postgraduate studies.
- My faithful friends for their motivation to achieve my goals.

*Ahmed Shokry*

# Abstract

This master thesis aims to investigate some Pathological issues of the FEM on static analysis of plane problems. Its objective is to analyze the results of the plane elements and find a way to improve them in simple and efficient way.

Four types of plane elements are being discussed; Constant Strain Triangle CST, Linear Strain Triangle LST, Bilinear Quadratic element Q4 and Quadratic Quadrature element Q8. Each type has its advantages and defects. Some locking problems as shear locking and compressibility locking are discussed too. They are the main reasons of the wrong results of the plane elements. How the choice of the numerical integration can affect the analysis and overcome the shear locking problem. An illustrative example is analyzed to understand the locking problems. Computer program for each example has been written and included in the appendices. The program used for the analysis is MatLab.

# Table of Contents

# Table of Figures

# List of Equations

# List of Tables

# List of notations

| | |
|---|---|
| $\sigma_x, \sigma_y$ | Normal stresses in direction of x and y respectively |
| $\tau_{xy}$ | Shear stresses in direction of plane x and normal to plane y |
| **x, y** | Cartesian coordinates |
| $F_x, F_y$ | Forces in direction of x and y respectively |
| **u, v** | Displacements in direction of x and y respectively |
| $\beta_i$ | Generalized coordinates |
| $\varepsilon_x, \varepsilon_y$ | Strain in direction of x and y respectively |
| $\gamma_{xy}$ | Shear strain in direction of plane x and normal to plane y |
| $M_x$ | Bending moment about x axis |
| $I_x$ | Second moment of inertia about x axis |
| $P_y$ | Concentrated load in y direction |
| **E** | Young's modulus of elasticity |
| **[E]** | Elasticity matrix |
| $[K^e]$ | Global stiffness matrix of an element |
| $[B^e]$ | Deformation matrix |
| **[N]** | Matrix of shape function |
| $\{\sigma\}$ | Stress vector |
| $\{\varepsilon\}$ | Strain vector |
| $\nu$ | Poisson's ratio |
| $\xi, \eta$ | Physical coordinates |
| $[B_c]$ | Deformation matrix of compatible elements |
| $[B_I]$ | Deformation matrix of incompatible elements |
| $\{d_c\}$ | Displacement vector of compatible elements |
| $\{d_I\}$ | Displacement vector of incompatible elements |
| **[J]** | Jacobian Matrix $\begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$ (2D) |

| | |
|---|---|
| **$|J|$** | Determinant of Jacobian matrix |
| **$U$** | Strain energy |
| **$\kappa$** | Curvature/Bulk modulus |
| **$t$** | Thickness of the beam |
| **$L/L_x$** | Length of the beam |
| **$r$** | Ratio between strain energy of the 4-node elements and that of the exact model |
| **$U_{quad}$** | Strain energy of 4-node elements |
| **$U_{beam}$** | Exact strain energy |
| **$\beta$** | Correction factor of strain energies (reciprocal of r) |
| **$K_\beta^e$** | Stiffness matrix integrated using weighted integration method |
| **$K_s^e$** | Stiffness matrix integrated using selective integration method |
| **$[\sigma_{Hyd}]$** | Hydrostatic stress |
| **$[\sigma']$** | Deviatoric stress |
| **${}^4C$** | Fourth-order stiffness tensor |
| **${}^4C^v$** | Volumetric part in Fourth-order stiffness tensor |
| **${}^4C^d$** | Deviatoric part in Fourth-order stiffness tensor |
| **$I$** | Fourth-order unit tensor |
| **$I^s$** | Symmetric fourth-order tensor |
| **$G$** | Shear modulus |
| **$B$** | Bulk modulus |

# List of abbreviations

| | |
|---|---|
| CST | Constant Strain Triangle, 1 |
| FEA | Finite Element Analysis, 1 |
| LST | Linear Strain Triangle, 1 |
| Q4 | Bilinear Quadrilateral Element, 1 |
| *Q6* | Improved Bilinear Quadrilateral Element, 31 |
| Q8 | Quadratic Quadrilateral Element, 1 |

# CHAPTER 1

INTRODCTION

# 1. Introduction

Since the development of the FEA in the early 40s had occurred, a great revolution happened in stress analysis problem. It was developed initially as a Computed Based Simulation Method for the analysis of aerospace structures, but then it is used in both structural and mechanical engineering; moreover in fluids, thermodynamics and electromagnetics [1, p. ii]. It can be used for the analysis of one dimensional element (bar element), 2D elements and solid elements. This report will focus on the analysis of 2D elements (plane elements).

FEM is a way of getting numerical solution to a specific problem. This solution is approximated to the exact one unless the problem is so simple that a convenient exact formula is already available [2, p. 1]. The accuracy of the results of the FEA depends mainly on selecting the type of element and the meshing size. As the mathematical description of the element becomes more complicated and detailed (Linear strain triangular element LST and quadratic quadrature element Q8), as the accuracy of the results improved. Also, it helped to fix the problems occurred in using simple elements (Constant strain triangular element CST and bilinear quadrature element Q4). Each type of element has its advantages and disadvantages. Studying their behavior will help in understanding and solving their defects.

However, the complicated mathematical description of a FE problem increases the time of computation and sometimes it is too difficult to find a solution. Reaching to the exact results in fast and efficient way is one of the most challenges that most engineers and mathematicians are facing.

E. L. Wilson and R. L. Taylor had developed a new element to fix the shear locking problem appeared Q4 element. This new element succeeded to get better results in more efficient and simple way. However, some defects appeared in this element that could be treated by using good mesh refinement.

Using other numerical integration to integrate the global stiffness matrix is a good solution to fix the shear locking problem. They approach the exact results with almost no defects. This could be done by using Gauss quadrature integration as discussed in chapter 4. However, insufficient choice of gauss points may cause spurious mode which has a really bad effect on the deformation of the element. This will be discussed in details in chapter 5.

Another problem affects the FEA is the volumetric locking problem. It occurs when using incompressible materials in the analysis like fluids. A good treatment for this problem is found in chapter 4.

The objective of this Master thesis is to focus on some pathological issues on FEM. How we could analyze the problem/errors in the FEM to reach a suitable solution with more accurate results. How the numerical analysis and modeling helps engineers to understand the behavior of the materials.

# CHAPTER 2

SOME DEFINITIONS & IMPORTANT REMARKS

# 2. Some Definitions & Important Remarks

This chapter provides some definitions of FE notations as *nodes*, *elements*, *local and global stiffness matrix* and *meshing size*. Some important remarks are mentioned to help understanding, analyzing and solving the problems of FEM.

## 2.1 Definition

A simple description of FEM is that it deals with cutting the structure into several elements, describing and analyzing each element in a simple way then reconnecting each element at "*nodes*" to form the whole structure again after analysis. This process is done using a set of algebraic equations which defined as "*Equilibrium Equations*" in stress analysis. The number of algebraic equations depends on the number of elements (the meshing size), that means a huge number of equations have to be calculated and a software program is mandatory [2, p. 1]. The division of the elements in the structure is called "the mesh". This element could be a bar element as in one dimensional structures i.e. truss structures, it could be triangle or rectangle as in 2D structures or it could be pentahedral or hexahedral as in 3D structures.

## 2.2 Remarks

Studying the behavior that the element is expected to display and the behavior that the elements are able to display is a good start to solve a FE problem. It will help in the selection the element types, size and shapes [2, p. 44]. The type of the element could be described by number of the nodes in it; you may have 4-noded element as in Q4 element and 8-noded element as in Q8 element. According to the geometry of the structure and the direction of loads, you can expect its deformed shape (maximum deflection), its bending moment and the location of the maximum and minimum stresses. From this prediction, the element type, the meshing size and shape could be selected. You could have benefits from the symmetry of the structure.

Then, the engineer turns to the software to analyze the problem. FEA is done according to these steps[1]:

1. Preparation of the FE model. The engineer should
   a. Define the properties of the material being used, modulus of elasticity and Poisson's ratio.
   b. Mention the dimension of the plane structure (if it is beam, the length, depth and thickness should be mentioned).
   c. Discretize the structure by dividing it into finite elements.
   d. Prescribe how the structure is loaded.
   e. Define the boundary conditions (how the structure is supported).
2. Preform the calculations. The software must
   a. Generate the stiffness matrix **k** of each element.
   b. Connects the elements together to formulate the global stiffness matrix **K**.
   c. Assemble the loads in load vector **R**.
   d. Solve the global equations $\boldsymbol{KD} = \boldsymbol{R}$ for the vector of displacements **D** which is the unknown.
   e. Impose the boundary conditions.
   f. Solve again the global equations mentioned in item d for **D**.
3. Now, it is easy to get the stresses in the elements and therefore the structure.

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + F_x = 0 \qquad \text{and} \qquad \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + F_y = 0$$

**Eq. 2-1** Equilibrium equations

Elasticity theory shows that if displacement and stress fields satisfy equilibrium equations, compatibility and boundary conditions on stress, then the obtained solution is exact. In FEM, the elements are based on polynomial displacement field. Then compatibility is satisfied within elements but equilibrium equations is not satisfied because Eq.2.1 are not satisfied at most points within the FE mode and boundary conditions on stress are not satisfied at most points on the boundary. But both of them are satisfied on an average sense. As the mesh is repeatedly refined, they approach satisfaction [2, p. 45].

[1] These procedures are done in the codes used in the analysis. You can check them in chapter 7 (appendices)

The local and global stiffness matrix, **k** and **K** respectively, are symmetric as a result of the linear relation between the applied loads and the resulting deformation. The following are some properties of k to avoid singularity [2, p. 25]:

- Diagonal coefficients of both of them are positive.
- Supports must be sufficient to prevent all possible rigid body motions.
- Structures may have singular K because it contains a mechanism[2].

[2] More details of this item can be found in chapter 4.

# CHAPTER 3

TYPES OF PLANE ELEMENTS

# 3. Types of Plane Element

In this chapter, four types of elements are discussed; Constant Strain Triangle CST, Linear Strain Triangle LST, Bilinear Quadrilateral Element Q4 and Quadratic Quadrilateral Element. An illustrative example is provided with full analysis to determine the limitations (advantages and disadvantages) of each type.

## 3.1 Constant Strain Triangle CST

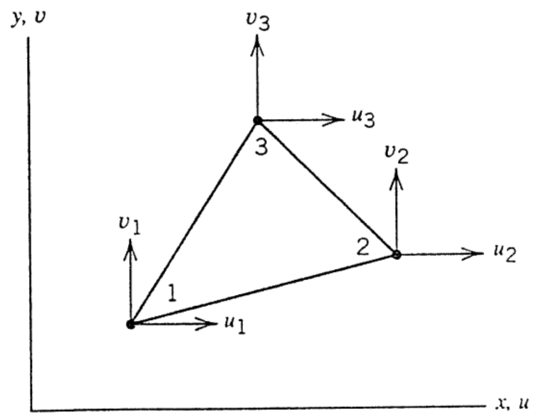


**Fig. 3-1** Constant Strain Triangle **[2, p. 46]**

Fig.3.1 shows the d.o.f. of a CST element. Perhaps, it is the earliest and simplest finite element. It can be used to discretize any complicated shape. Its displacement field in terms of generalized coordinate $\boldsymbol{\beta_i}$ is:

$$u = \beta_1 + \beta_2 x + \beta_3 y$$
$$v = \beta_4 + \beta_5 x + \beta_6 y$$

**Eq. 3-1** Displacement field of CST

And from Eq.3.1, we can get the strain field:

$$\varepsilon_x = \beta_2$$
$$\varepsilon_y = \beta_6$$
$$\gamma_{xy} = \beta_3 + \beta_5$$

**Eq. 3-2** Strain field of CST

From Eq.3.2, we can deduce that the strain is constant along the element; therefore it is called "*Constant Strain Triangle*". It is also called "*Linear Triangle*" due to the linear relation of the displacement field in x and y.

## Illustrative Example

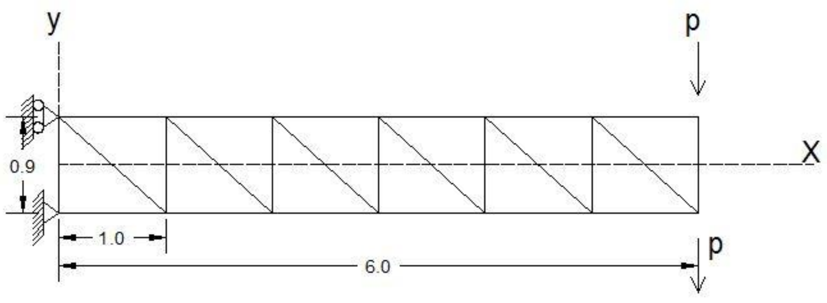


**Fig. 3-2** Cantilever Beam modeled by CST

Fig.3.2 shows a cantilever beam of length 6 m, height 0.9 m and its thickness is 0.3 m. The modulus of Elasticity of its material is 210 KN/m$^2$ and Poisson's ratio is 0.3. This cantilever beam is loaded by 50 KN at its free end distributed on two nodes each of 25 KN (P = 25 KN).

According to the classical beam theory [3, p. 396], the normal stress is given by

$$\sigma_x = \frac{M_x y}{I_x}$$

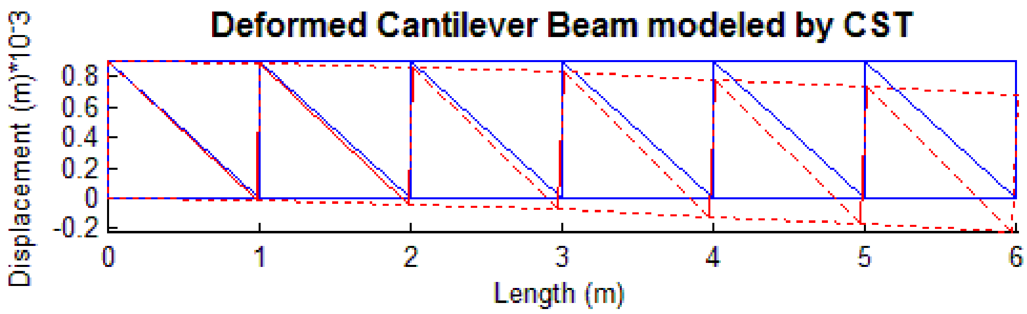
**Eq. 3-3** Normal Stress according to the beam theory

While the deflection at the free end is given by [3, p. 411]

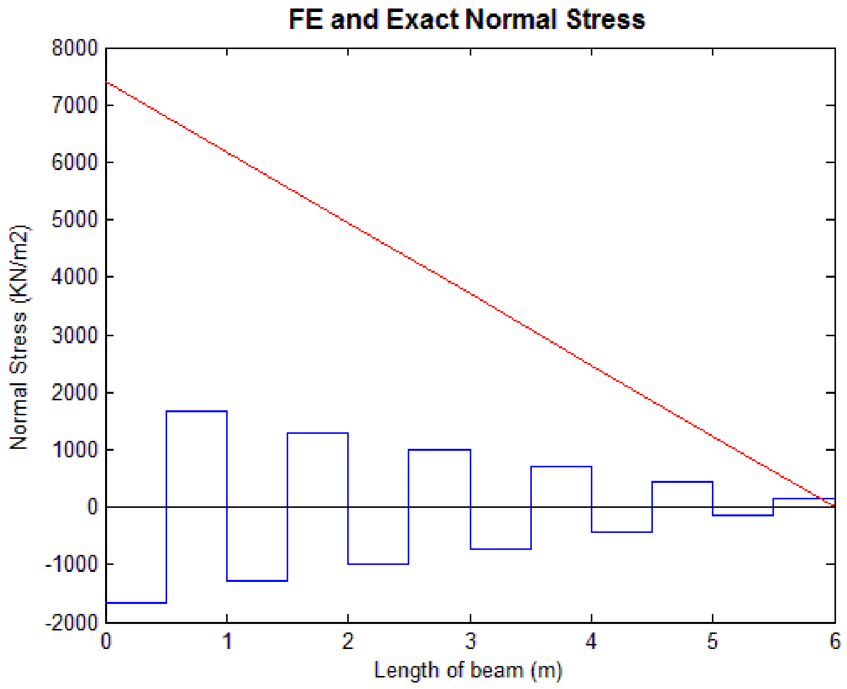
$$v = \frac{-P_y L^3}{3EI_x}$$

**Eq. 3-4** Deflection of beam according to classical beam theory

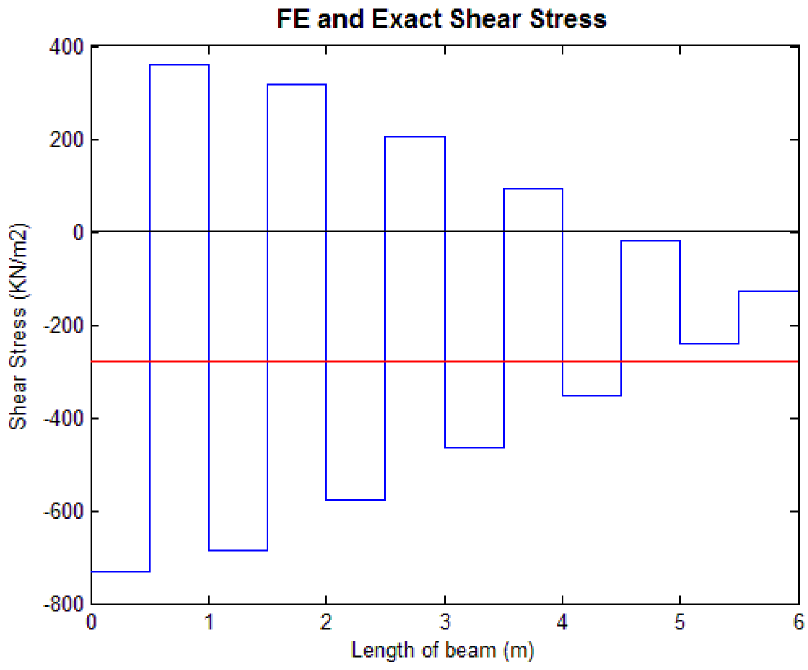
By substituting the properties of the beam in Eq.3.3, we get the exact normal stress which is 7404 KN/m$^2$ and substituting in Eq.3.4 gives the deflection which is 9.406*10$^{-4}$ m. More details about the exact analysis of the cantilever beam can be found in a script Appendix A.



**(a)** $v = -2.22*10^{-4}$ m.[3]



(b) $\sigma_x = -1656$ KN/m$^2$ [4]



(c)

[3] The deflection shown in Fig.3.3a is multiplied by 1000.
[4] The red line refers to the exact results while the blue line refers the FE results (This implies on all the graphs except for the deformation).

**Fig. 3-3** Analysis of CST[5]

The deflection of the beam shown in Fig.3.3a confirms the linearity of the displacement field given in Eq.3.1. It is deformed as broken lines connected together. Also, it is shown from Fig.3.3b that the first element has the max normal stress of -1656N/m$^2$ in the FE model compared to 7407KN/m$^2$ in the exact model. The position of the centroid for each element varies from one to another; one element is in negative y axis and the other is in positive y axis, which is a reason of the wave pattern shape appeared in both the Normal and Shear Stresses; that is; the first element is exposing to compressive stress because its centroid is located at in the negative y, while the centroid of the second element is exposed to tensile stress because its centroid is located at the positive y.

The big gap between the values of stresses, shown in Fig.3.3b&c, between the FE model and exact model is a result of CST inability to represent a $\boldsymbol{\varepsilon_x}$ that varies linearly with y (that is clarified clearly in Fig.3.3a). CST only gives good results in region of FE model where there is a little strain gradient, otherwise it doesn’t work. Despite this defect, CST gives better results in repeatedly refined mesh [2, p. 47].

## 3.2 Linear Strain Triangle LST

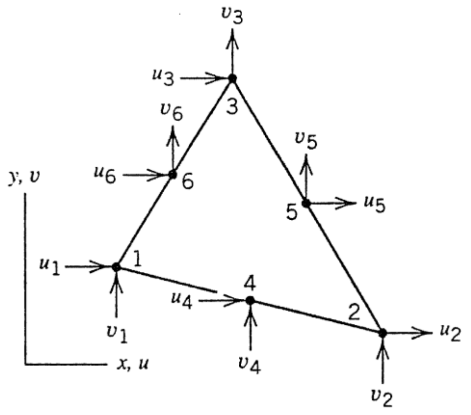


Fig. 3-4 Linear Strain Triangle **[2, p. 48]**

LST differs from CST by its six nodes, as shown in Fig.3.4. Each node has two d.o.f, so there are 12 d.o.f in this element.Its displacement field is

[5].The MatLab script of the FE analysis is found in Appendix B, section 7.2.1

$$u = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 x^2 + \beta_5 xy + \beta_6 y^2$$
$$v = \beta_7 + \beta_8 x + \beta_9 y + \beta_{10} x^2 + \beta_{11} xy + \beta_{12} y^2$$

**Eq. 3-5** Displacement field of LST

From Eq. 3.5, the strain field will be:

$$\varepsilon_x = \beta_2 + 2\beta_4 x + \beta_5 y$$
$$\varepsilon_y = \beta_9 + 2\beta_{11} x + \beta_{12} y$$
$$\gamma_{xy} = (\beta_3 + \beta_8) + (\beta_5 + 2\beta_{10})x + (2\beta_6 + \beta_{11})y$$

**Eq. 3-6** Strain field of LST

Eq.3.6 shows that there is a linear relation between the strain field in x & y and their corresponding axis, that is why it is called Linear Strain Triangle. It may also called Quadratic Triangular Element due to the quadratic relation of the displacement field in x and y.

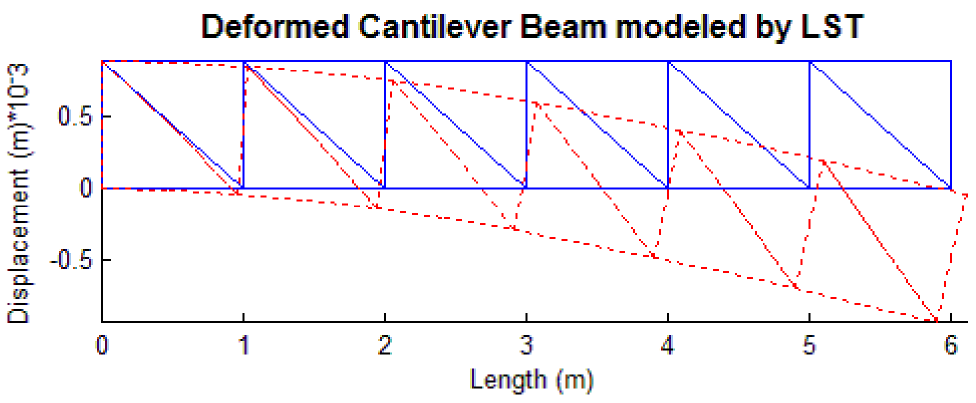
**Illustrative example**

The cantilever beam shown in Fig.3.2 is now modeled by LST[6] with the same material and cross section properties used before.

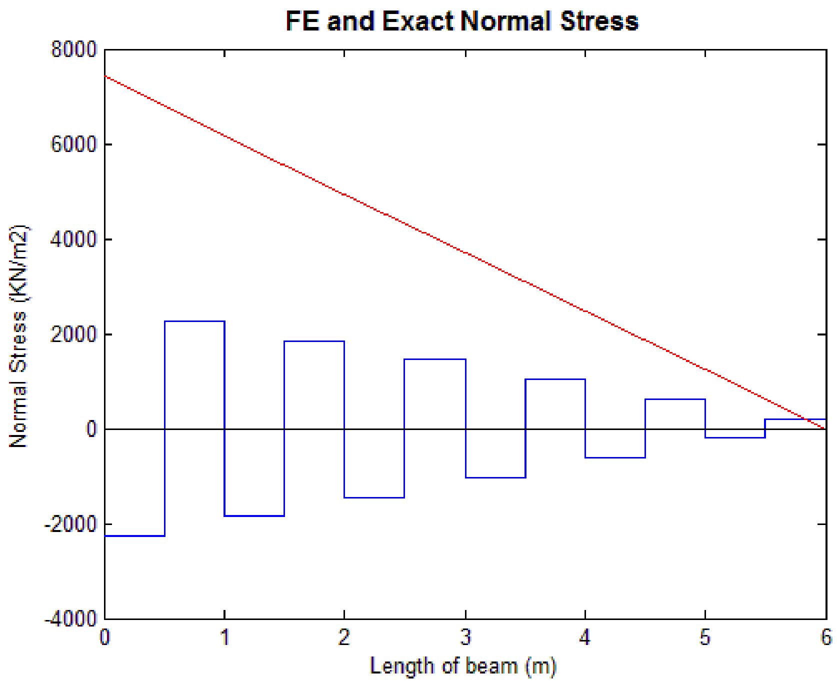
In Fig.3.6a, the deformed cantilever beam confirms the nonlinearity of the displacement field of LST. It approaches the exact value of the deformation. The FE normal stress shown in Fig.3.5b is much better than in CST even though it is far away from the correct result. Fig.3.5c shows linear relation of the shear stress rather than the wave pattern shown in Fig.3c in CST.

From this analysis, we can conclude that LST has all the capabilities of CST and more as it can represent a linear relation between $\boldsymbol{\varepsilon_x}$ and y. LST gives more accurate results of deformations and stresses. This is clearly obvious by comparing the results in Fig.3.3 and Fig. 3.5

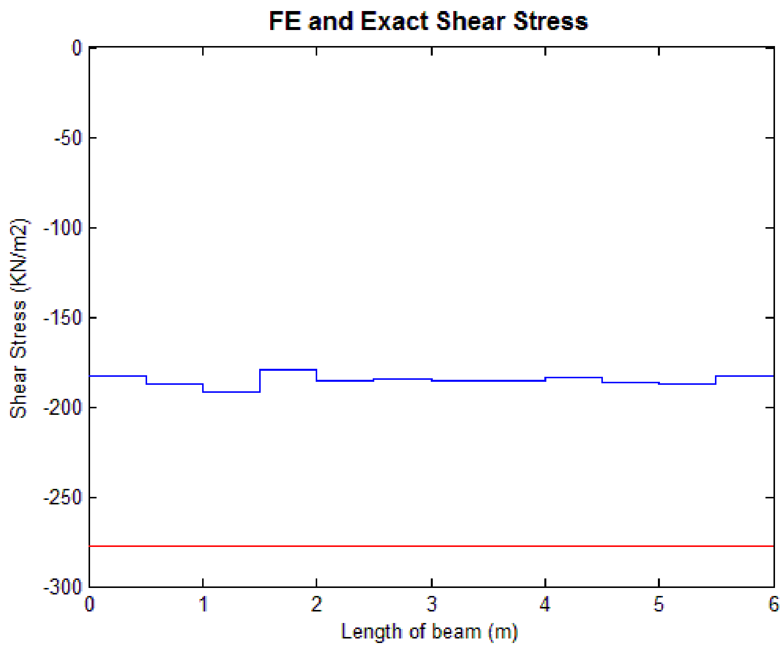
[6] More details of the script can be found in Appendix B, section 7.2.2.



**(a)** $v = -9.416*10^{-4}$ m



(b) $\sigma_x = 2265$ KN/m$^2$



(c)

**Fig. 3-5** Analysis of LST

## 3.3 Bilinear Quadrilateral Elements Q4

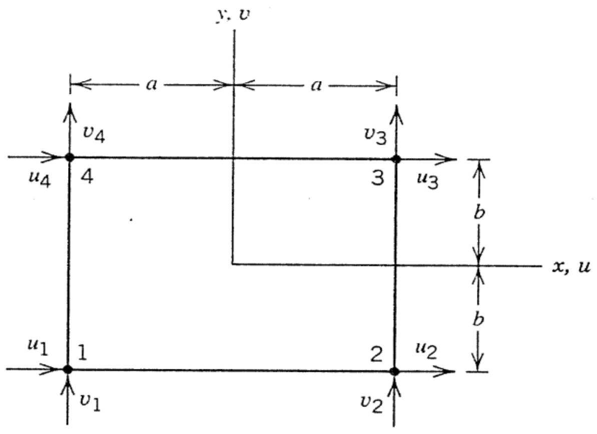


**Fig. 3-6** A bilinear quadrilateral element **[2, p. 49]**

The Q4 element is a quadrilateral that has four nodes. Each of them has two d.o.f. Its displacement field is given by:

$$u = \beta_1 + \beta_2 x + \beta_3 y + \beta_4 xy$$
$$v = \beta_5 + \beta_6 x + \beta_7 y + \beta_8 xy$$

**Eq. 3-7** Displacement field of Q4 element

Eq.3.7 shows that both forms of u and v are the product of two linear polynomial ($c_1$+$c_2$x) and ($c_3$+$c_4$y) so the name Bilinear Quadrilateral arises [2, p. 48]. From this equation we can get the strain field:

$$\varepsilon_x = \beta_2 + \beta_4 y$$
$$\varepsilon_y = \beta_7 + \beta_8 x$$
$$\gamma_{xy} = (\beta_3 + \beta_6) + \beta_4 x + \beta_8 y$$

**Eq. 3-8** Strain field of Q4 element

Q4 element is very popular among FE users, so it is going to be discussed in further details. Two beams are being analyzed; that is, a cantilever beam with one concentrated load at its edge and a simple beam with one concentrated load at the mid-span. Each one will be divided by three different ways (three meshing size), each will have different load distributing along the nodes. Tables 3.1 to 3.4 define a code for each beam, meshing size and way of distributing the load.

| Code | Identification |
|---|---|
| **Beam_X1** | The exact analysis |
| **Beam_X2** | The FE model of Simple Beam |
| **Beam_X2_No.1** | No.1 a number refers to the meshing size which is going to be used |
| **Beam_X2_No.1_No.2** | No.2 a number refers to the way of distributing the load along the nodes |

**Table 3-1** Example of the code identification of the beam in Q4

| Beam_X | Identification |
|---|---|
| **SimpleBeam_A1** | Exact analysis of Simple Beam (1 concentrated load) |
| **SimpleBeam_A2** | FE model of Simple Beam (1 concentrated load) |
| **CantileverBeam_C1** | Exact analysis of Cantilever Beam (1 concentrated load at its free end) |
| **CantileverBeam_C2** | FE model of Cantilever Beam (1 concentrated load at its free end) |

**Table 3-2** Identification of the type of beam in Q4

| No.1 | Identification | | |
|---|---|---|---|
| | No. of elements in x direction | No. of elements in y direction | Total no. of elements |
| **01** | 6 | 1 | 6 |
| **02** | 12 | 4 | 48 |
| **03** | 60 | 18 | 1080 |

**Table 3-3** Identification of the mesh size in Q4

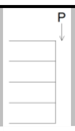
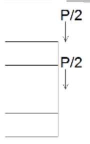
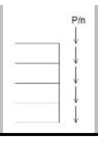
| No.2 | Identification | |
|---|---|---|
| **01** |  | **Fig. 3-7** Concentrated load on one node |
| **02** |  | **Fig. 3-8** Concentrated load is divided on two nodes |
| **03** |  | **Fig. 3-9** Concentrated load is divided along the elements |

**Table 3-4** Identification of the load distributing in Q4

## Distribution of the load

As mentioned before, FEM is the division of the beam into elements. Each element has number of nodes; in this section it is four-node element. Axiomatically, the node exposed to the nodal force has the most displacement and the elements contain this node exposed to the maximum stresses. The FE model considers the nodal force is carried only on this node, while in reality each node surrounding the load (especially those under the nodal force) carries a portion of the load. So we have to reach an optimum way of distributing the load to get a result that resembles the reality.

Fig.3.10 shows a simple beam used to study the effect of distributing the concentrated load on the mid span. This simple beam is discretized to 1080 elements as in Fig. 3.11.

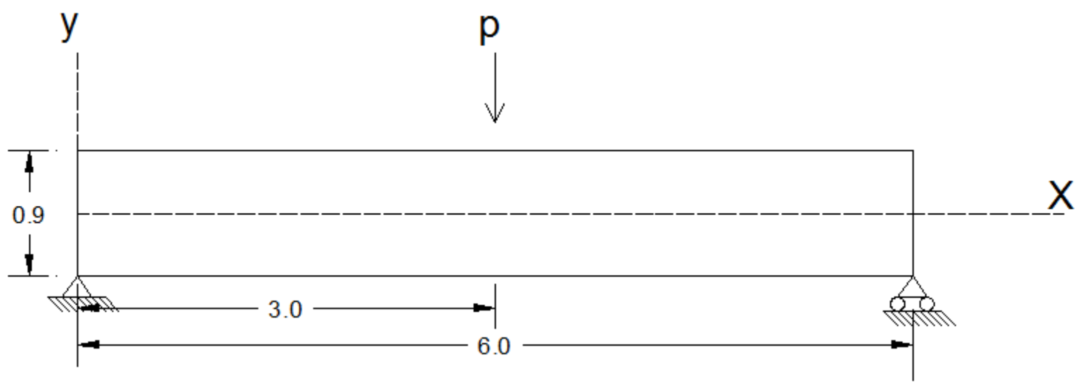


**Fig. 3-10** Simple beam

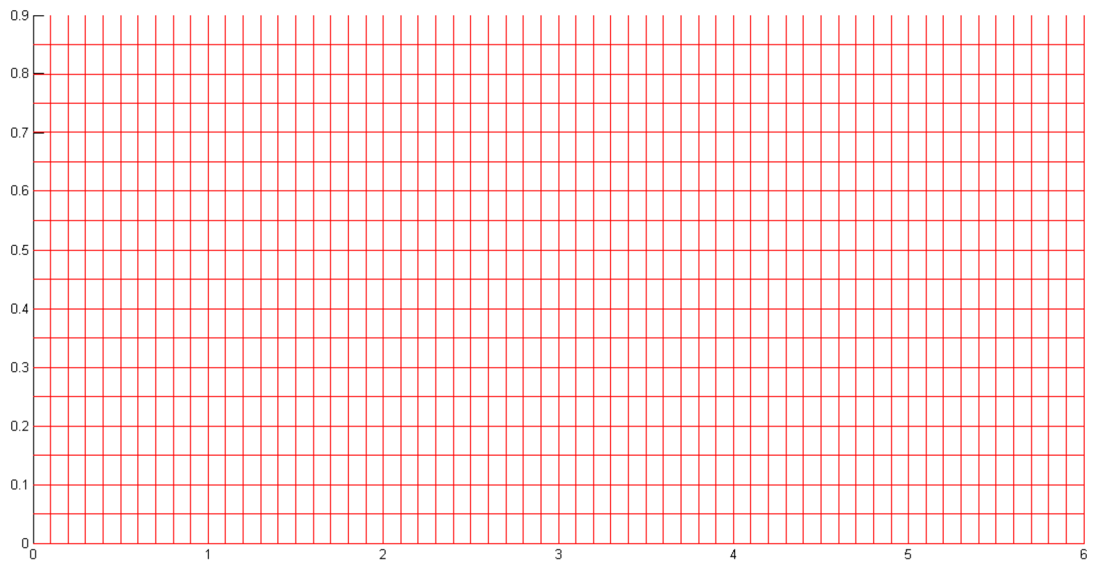


**Fig. 3-11** Meshing size of Simple Beam_A2_03

Fig.3.12 shows the effect of loading the concentrated load (50 KN in this example) on one node on the normal stress distribution of the beam. A big drop can be noticed in the FE model at the node where the load is acting. Fig.3.13 shows better results of the normal stresses when the concentrated load is divided between the node where the load is acting on and its corresponding at the bottom edge of the beam. However, Fig.3.14 shows much better results (no drop at the peak) where the concentrated load is distributed on the nodes below it. The normal stress distribution is almost coincides with the exact distribution.

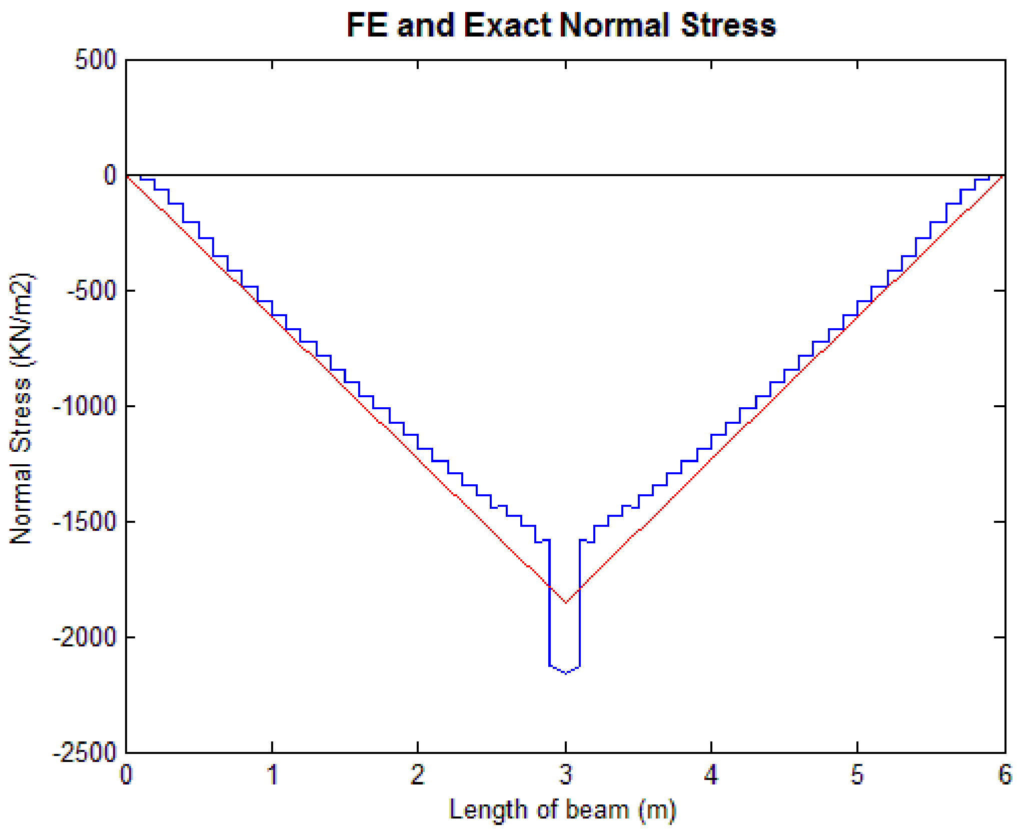


**Fig. 3-12** Normal Stress in SimpleBeam_A2_05_01

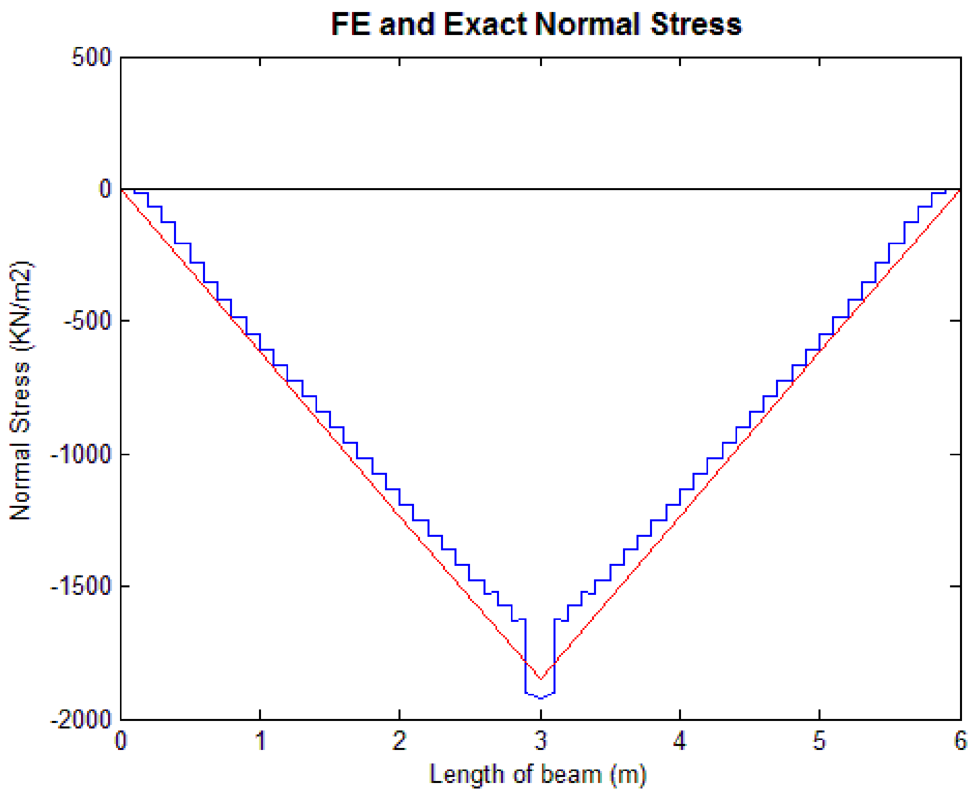


**Fig. 3-13** Normal Stress in SimpleBeam_A2_05_02

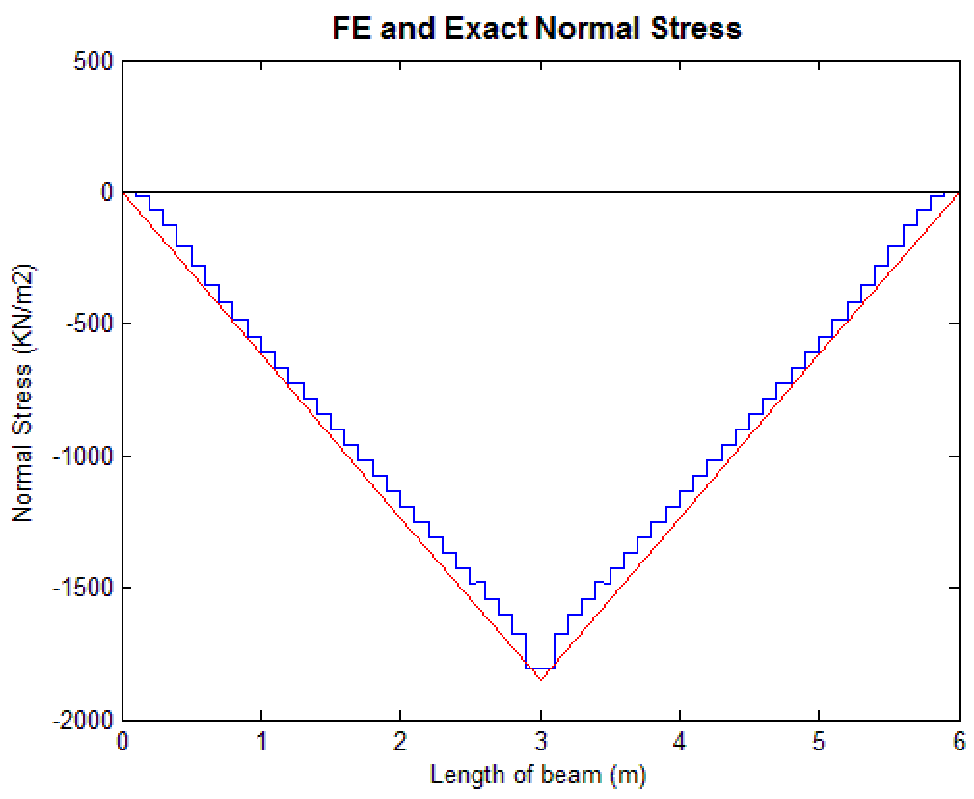


**Fig. 3-14** Normal Stress in SimpleBeam_A2_05_03

It is obvious that the optimum way to distribute the nodal force is to divide it on the nodes below. This distribution will be used as default in the following analyses.

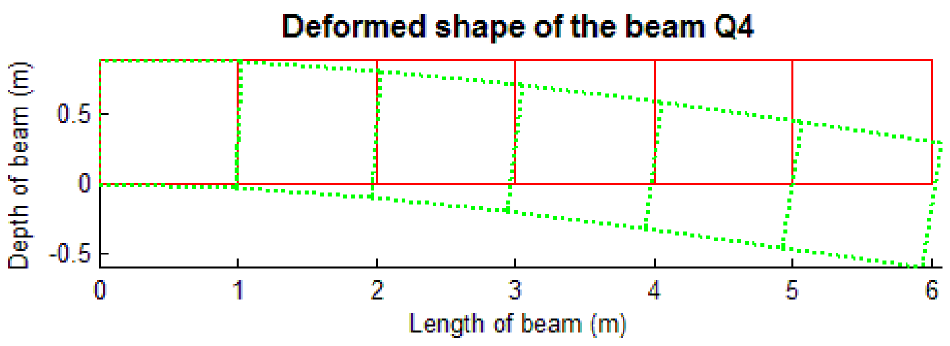
## Meshing Size

As meshing size becomes smaller, the FE model gives accurate results. However the software machine takes longer time to solve the problem. Analyzing the behavior of Q4 element will help to recognize its defects and how to solve them to reach the optimum meshing size. We are comparing three different meshing sizes to the exact value with respect to the normal and shear stresses.

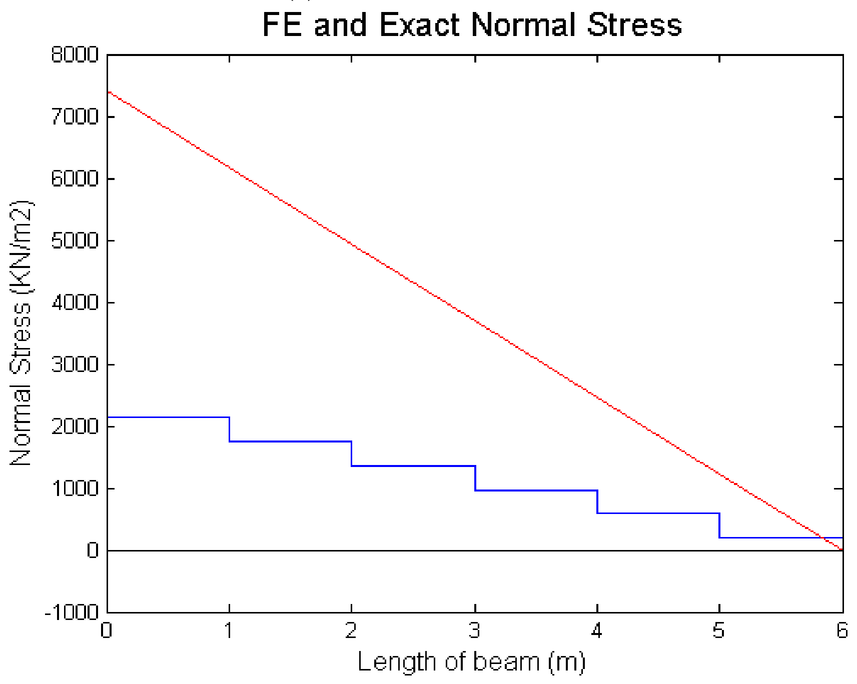
## Illustrative Example

The same cantilever beam used in the previous sections is now modeled by Q4 element. Three meshing size were used to show the effect of increasing the mesh in the analysis[7].

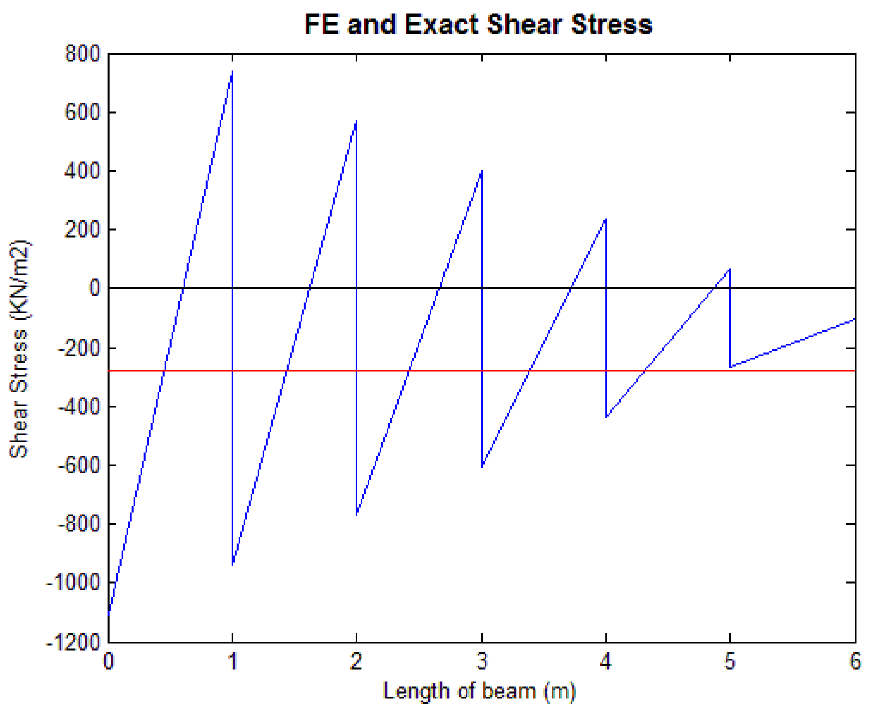
[7] More details of the script can be found in Appendix C, section 7.3.



**(a)** $v = -6.073*10^{-4}$ m

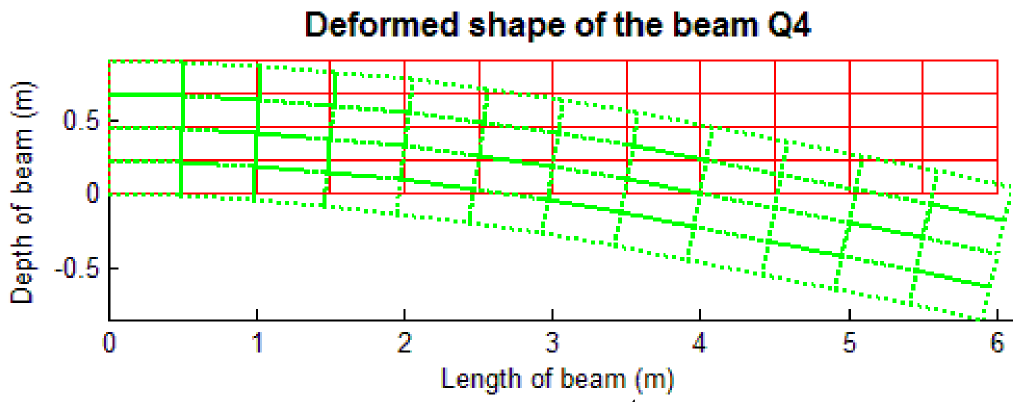


(b) $\sigma_x = 2133$

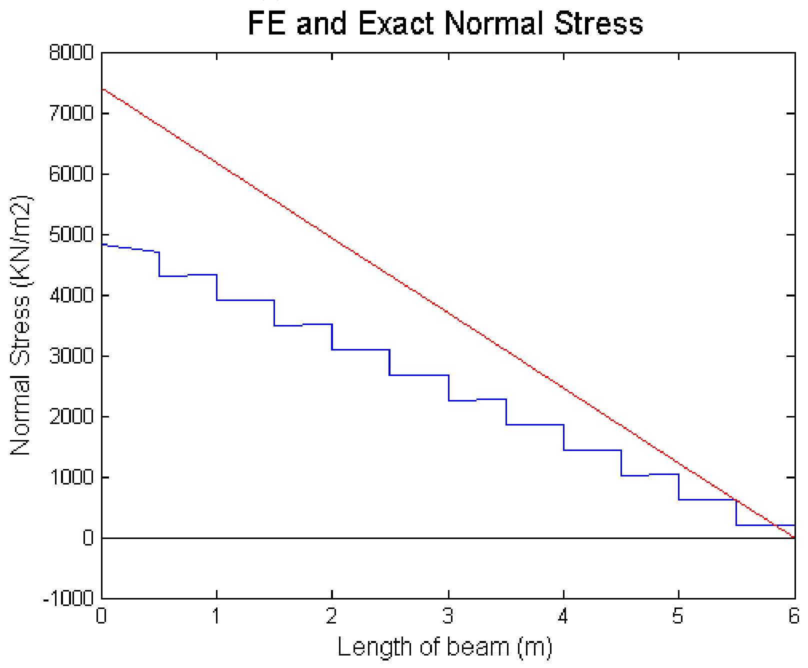


(c)

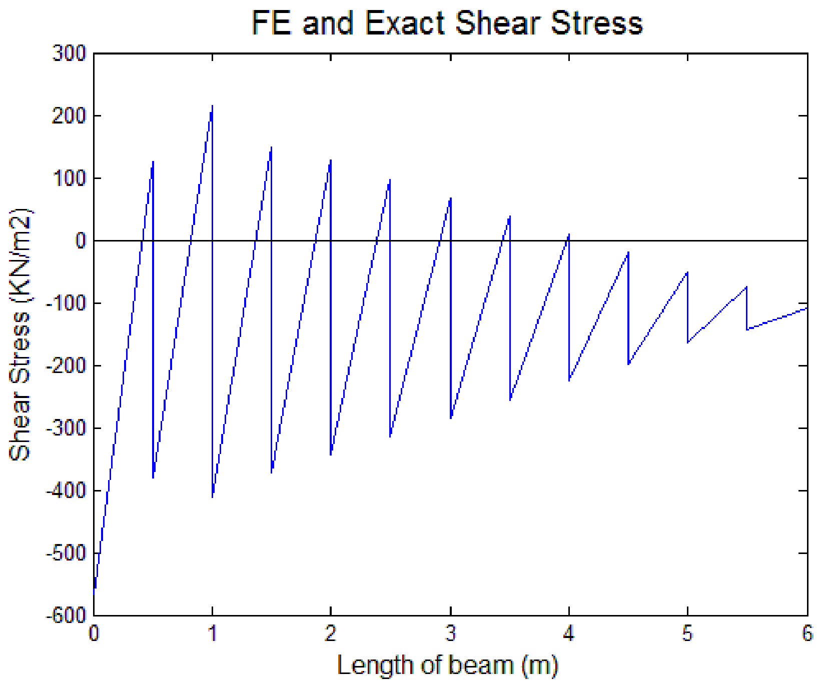
**Fig. 3-15** Analysis of CantileverBeam_C2_01_03_Q4 modeled by Q4 element



(a) $v = -8.449 \times 10^{-4}$ m

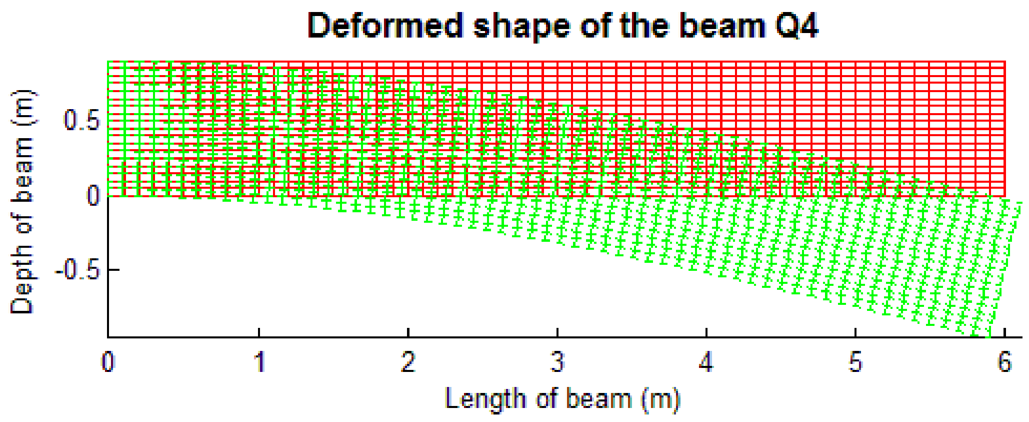


(b) $\sigma_x = 4830$

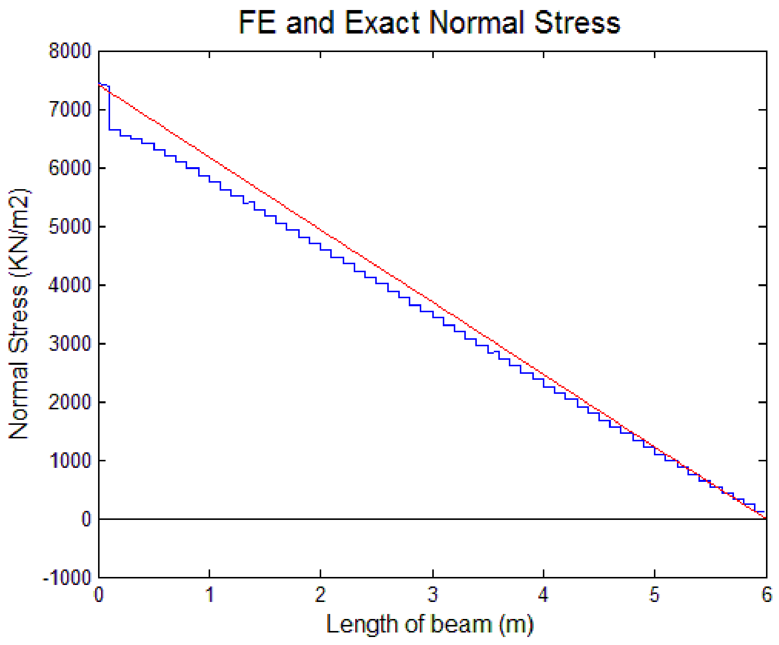


(c)

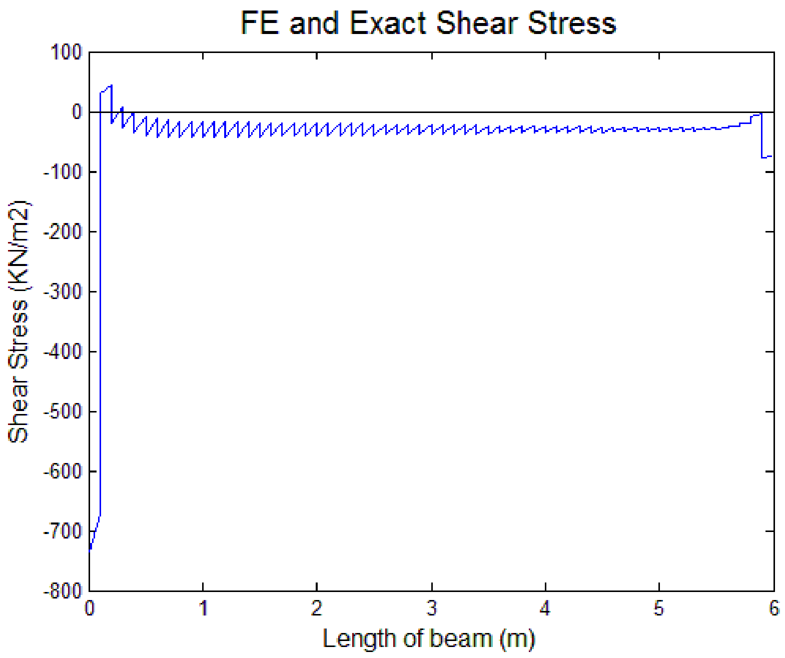
**Fig. 3-16 Analysis of** CantileverBeam_C2_03_03_Q4 modeled by Q4 element



(a) v = $-9.49*10^{-4}$



(b) $\sigma_x$ = 7424



(c)

**Fig. 3-17** Analysis of CantileverBeam_C2_05_03_Q4 modeled by Q4 element

The deformed shape of the cantilever beam shown in Fig.3.15a clarifies the linear relation of the displacement field and x & y (Eq.3.7). It also shows that $\boldsymbol{\varepsilon_x}$ is independent of x (Notice $\boldsymbol{\varepsilon_x}$ is constant along the length of the beam). This means that Q4 cannot model a state of pure bending. This important aspect of the element behavior will be discussed in more details in chapter 4. In Fig.3.15b, the normal stress of the FE model is 2133KN/m$^2$ which is too small compared to the exact value 7407KN/m$^2$. The shear stress[8] for the same model is -731KN/m$^2$ which is too large compared to the corresponding exact value - 277.7KN/m$^2$. By increasing the meshing size, Fig.18a & 18b, the deflection and normal stress are almost the same and the shear stress in Fig.18c approaches its exact value which is zero[9]. From this analysis, it is obvious that the normal stress of the FE model is usually less than that of the exact value, while the shear stress is more than the exact value. This is due to *shear locking;* one of the important defects of Q4 elements. More details about this problem are discussed in section 4.1.

**Summary of Q4 element**

To satisfy the equilibrium (Eq2.1), a state of constant strains should prevail. In other words, $\beta_4$ & $\beta_8$ in the displacement field (Eq.3.7) should be equal to zero. Despite this problem, Q4 element gives better results than CST as it converges properly with mesh refinement although CST satisfies the equilibrium equations [2, p. 50].

## 3.4 Quadratic Quadrilateral Element Q8

The Q8 element is a quadrilateral element with 8 nodes. Each node has two d.o.f. as shown in Fig.3.18. Its displacement field is:

$$\begin{aligned} u &= \beta_1 + \beta_2 x + \beta_3 y + \beta_4 x^2 + \beta_5 xy + \beta_6 y^2 + \beta_7 x^2 y + \beta_8 xy^2 \\ v &= \beta_9 + \beta_{10} x + \beta_{11} y + \beta_{12} x^2 + \beta_{13} xy + \beta_{14} y^2 + \beta_{15} x^2 y \\ &\quad + \beta_{16} xy^2 \end{aligned}$$

**Eq. 3-9** Displacement Field of Q8

[8] For the model in Fig.3.15c, the shear stress is calculated at the centroid of the elements.
[9] The shear stress is calculated for the models in Fig.3.16c & 3.17c at the top elements

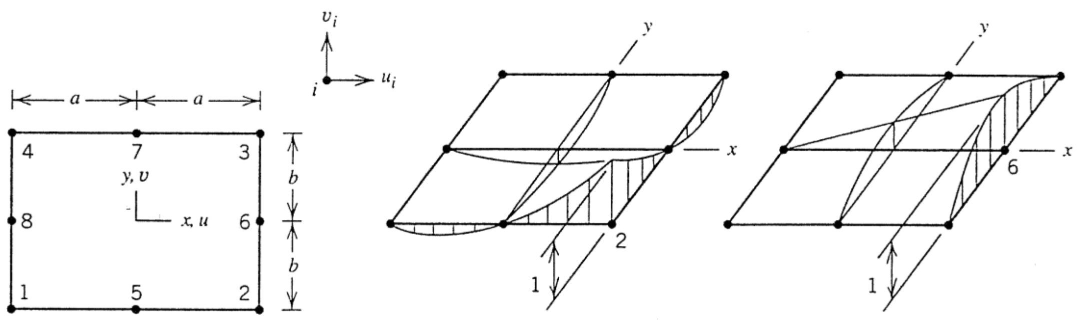


**Fig. 3-18** Quadratic Quadrilateral Element Q8 with its shape functions[10]

From the displacement field, the strain field can be calculated as:

$$\begin{aligned}\varepsilon_x &= \beta_2 + 2\beta_4 x + \beta_5 + 2\beta_7 xy + \beta_8 y^2 \\ \varepsilon_y &= \beta_{11} + \beta_{13} + 2\beta_{14} y + \beta_{15} x^2 + 2\beta_{16} xy \\ \gamma_{xy} &= (\beta_3 + \beta_{10}) + (\beta_5 + 2\beta_{12})x + (2\beta_6 + \beta_{13})y + \beta_7 x^2 \\ &\quad + 2(\beta_8 + \beta_{15})xy + \beta_{16} y^2\end{aligned}$$

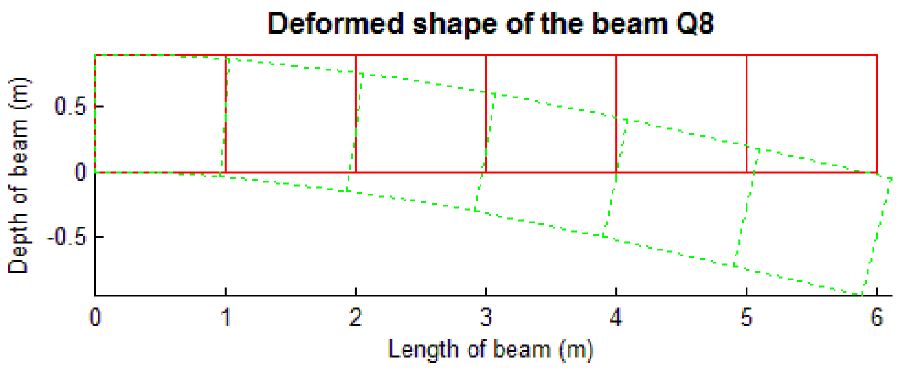
**Eq. 3-10** Strain Field of Q8

**Illustrative Example**

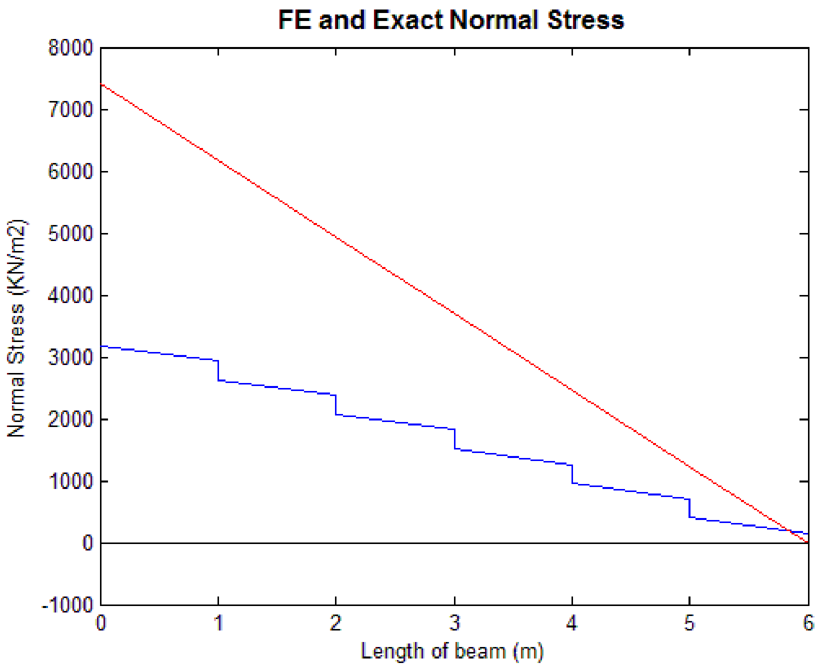
The same cantilever beam is analyzed using Q8 element with the same criteria used to analyze Q4[11]. The displacement field in Eq.3.9 shows that there is a quadratic relation between the displacement and x & y, so it is called Quadratic Quadrilateral Element. The deformed shape of the cantilever beam shown in Fig.3.19 clarifies this relation; it has more flexible deformation (nonlinear) than that in Fig. 3.16a. The distribution of the normal stresses in Fig.3.19b to 3.21b is getting closer to that of the exact distribution. The shear stress shown in Fig.3.19c to 3.21c is no longer a wave pattern (Fig.3.15c to 3.17c). Generally, Table 3.5 proves that the Q8 element gives better result compared to Q4 element especially for less mesh refinement for both shear and normal stresses.

Unlike Q4 element, Q8 element can represent all states of constant strain and states of pure bending [2, p. 51]. It is shown in the defamation modes in Fig3.19a to 3.21a, the relation is nonlinear between the displacement and x & y axis.

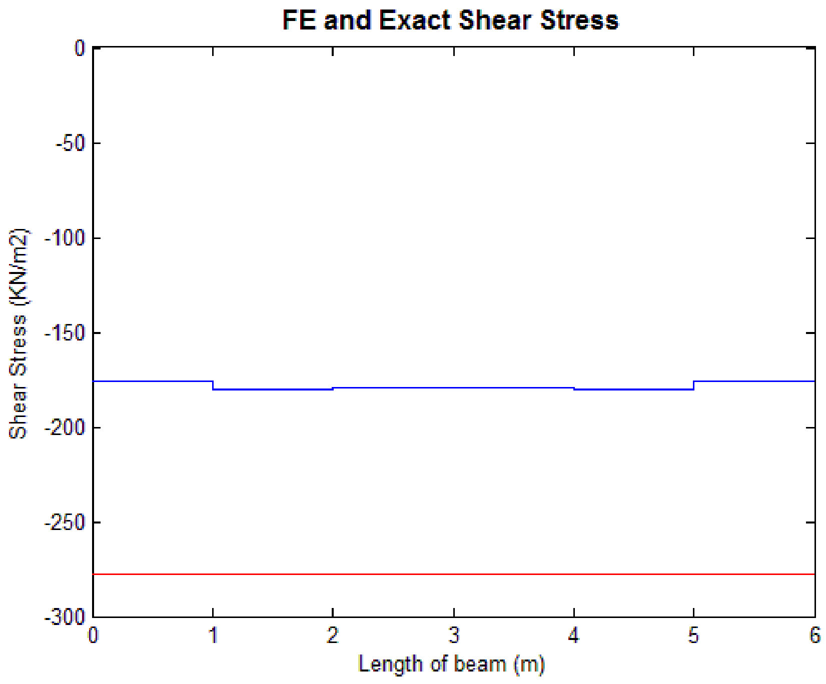
[10] The displacement shown is for imagination only, the real one occurs in x-y plane.
[11] The MatLab code used for this model discussed in Appendix D (Section 7.4)



$v = -9.51*10^{-4}$

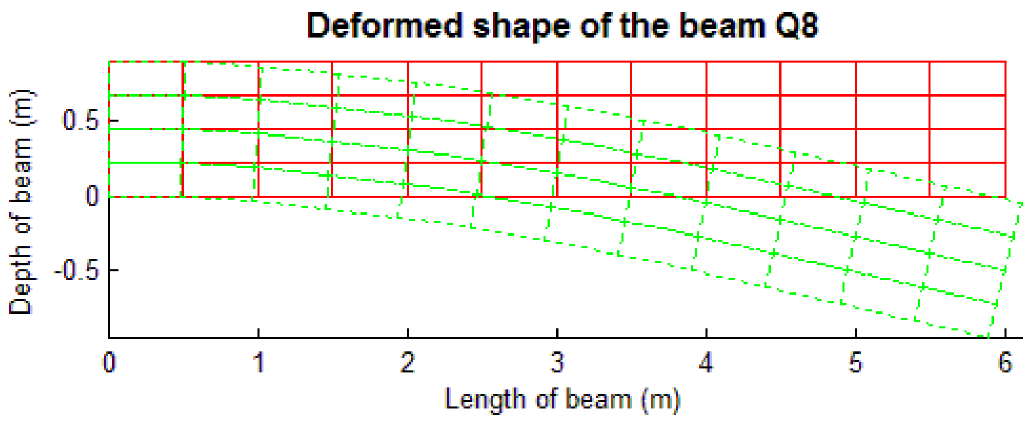


(b) $\sigma_x = 3183$

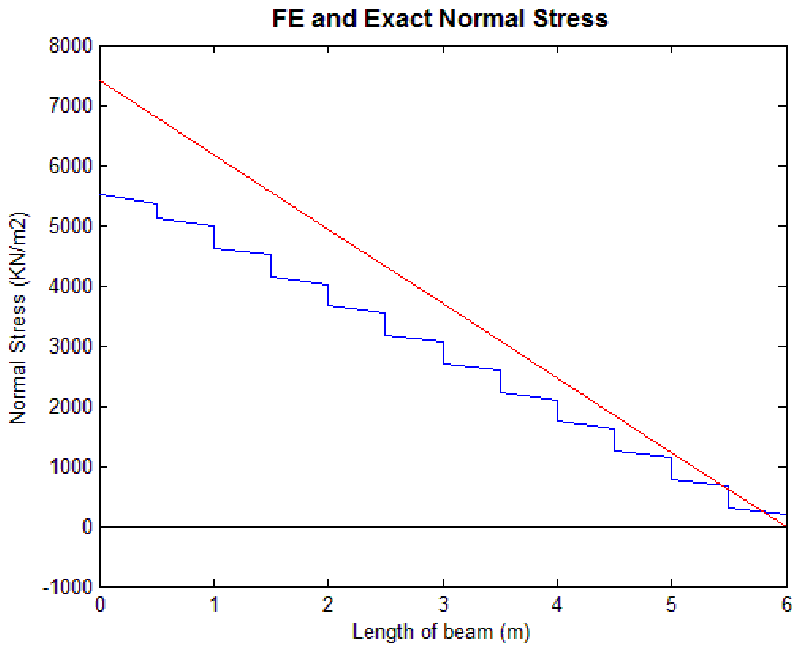


(c)

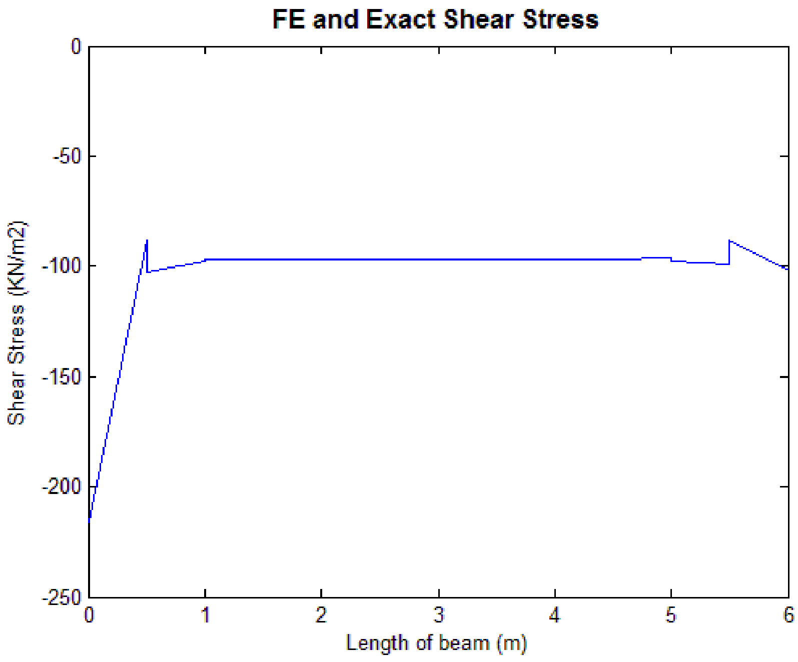
**Fig. 3-19** Analysis of CantileverBeam_C2_01_03_Q8 modeled by Q8 element



$v = -9.53*10^{-4}$

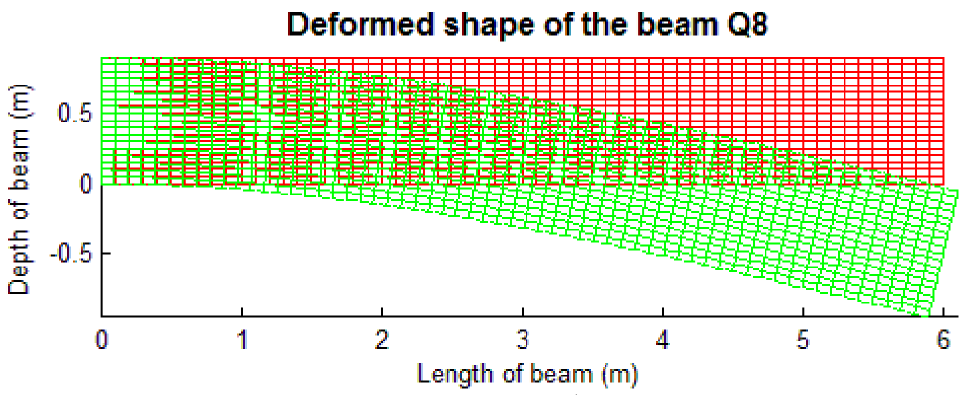


(b) $\sigma_x = 5052$

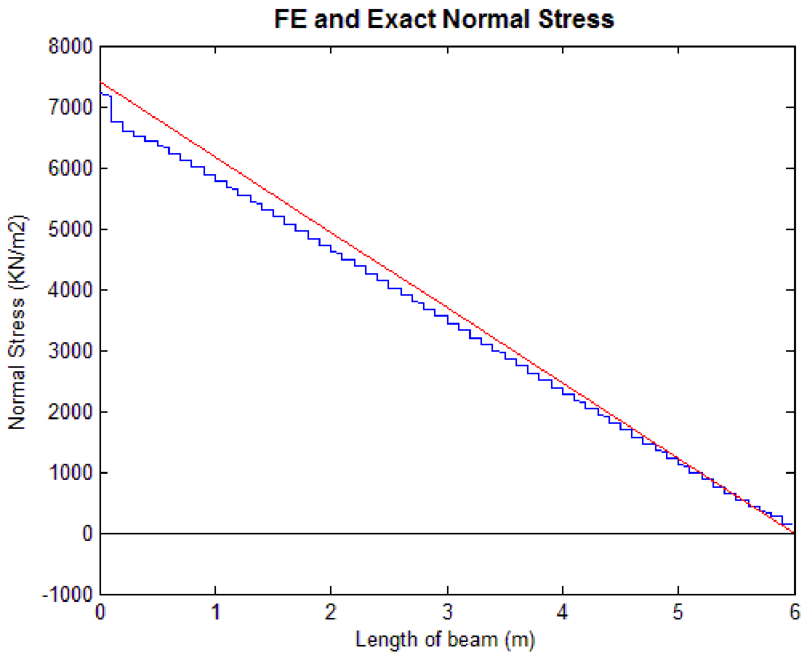


(c)

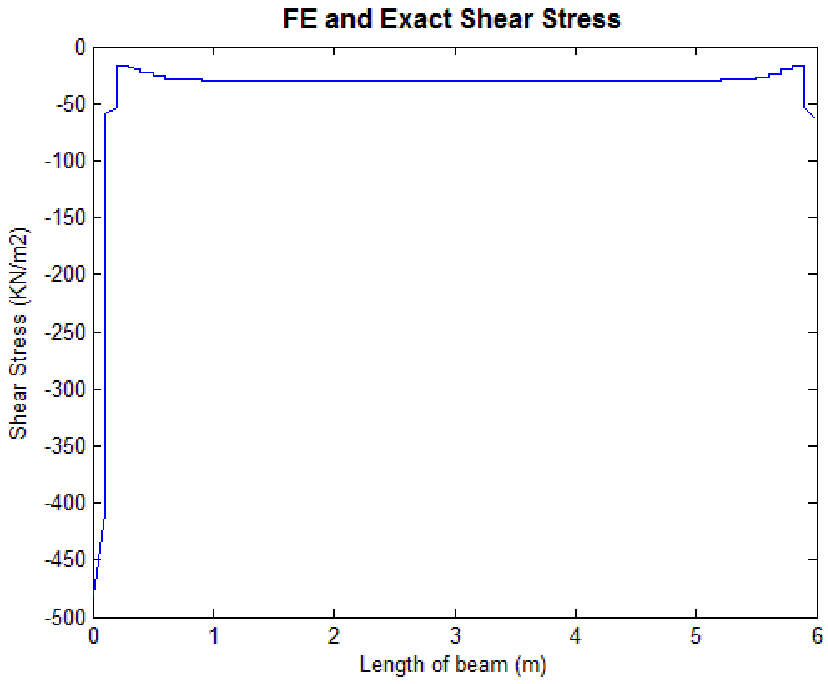
**Fig. 3-20** Analysis of CantileverBeam_C2_03_03_Q8 modeled by Q8



(a) v = -9.55*$10^{-4}$ m



(b) $\sigma_x$ = 7249 KN/$m^2$



(c)

**Fig. 3-21** Analysis of CantileverBeam_C2_05_03_Q8 modeled by Q8 element

| | | Exact | Q4 element | Q8 element |
|---|---|---|---|---|
| **Normal stress ($KN/m^2$)** | 6 elements | 7407 | 2133 | 3183 |
| | 60 elements | | 4830 | 5052 |
| | 1080 elements | | 7424 | 7249 |
| **Deflection (m)** | 6 elements | $-9.406*10^{-4}$ | $-6.073*10^{-4}$ | $-9.51*10^{-4}$ |
| | 60 elements | | $-8.449*10^{-4}$ | $-9.53*10^{-4}$ |
| | 1080 elements | | $-9.49*10^{-4}$ | $-9.55*10^{-4}$ |

**Table 3-5** Comparison of analysis between Q4 and Q8 elements

## 3.5 Remarks

Some remarks should be mentioned about the stress analysis done in the previous subsections. FEM solves first for d.o.f. then use them to get the stresses. In plane stress problems, we get the global stiffness matrix of each element (Eq.3.11) to get the displacement (Eq.3.12). Then we get the strain (Eq.3.13) and finally the stresses (Eq.3.15).

In the previous analysis for the stresses – as in Fig.3.16b for example- we can notice that the stresses are discontinuous between the elements, while the deformation in Fig.3.16a is more accurate. This behavior is independent to the type of element. Displacements are more accurate than stresses because stresses are proportional to strains (Eq.3.15) while strains are derivatives of displacements (Eq.13&14). Differentiations bring out differences between functions. For example, the two functions $y = e^x$ and $y = 1 + x$ are very similar over the range $0 < x < 0.2$, but the first derivatives are $y' = e^x$ and $y' = 1$ are different and the second derivatives $y'' = e^x$ and $y'' = 0$ are very different. This example can be implied on the elements. Elements have similar displacement field. To get the stresses, they have to be differentiated which leads to stress discontinuity [2, p. 28&29].

$$[\boldsymbol{K^e}] = \int_{v_e} [\boldsymbol{B}^{(e)}]^T [\boldsymbol{E}][\boldsymbol{B}^{(e)}]\, dV_e$$

**Eq. 3-11** Global Stiffness matrix of an element

$$[\boldsymbol{F}] = [\boldsymbol{K}][\boldsymbol{U}]$$

**Eq. 3-12** Relation between Force and Displacement

$$\{\boldsymbol{\varepsilon}\} = [\boldsymbol{B}][\boldsymbol{d}]$$

**Eq. 3-13** Strain of an element

$$[\boldsymbol{B}] = [\partial_\varepsilon][\boldsymbol{N}]$$ **Eq. 3-14** Deformation matrix

$$\{\boldsymbol{\sigma}\} = [\boldsymbol{E}]\{\boldsymbol{\varepsilon}\}$$ **Eq. 3-15** Stresses

# CHAPTER 4

LOCKING PROBLEMS

# 4. Locking Problems

This chapter discusses the *shear* and *volumetric locking*. How they affect the analysis and how we can solve them to reach much better results. Definitions of volumetric and deviatoric stresses are also included. Illustrative example is mentioned to quite understand the problem and its solution.

## 4.1 Definition

Locking effect is used by engineers to describe the case where the FE computations produce smaller displacements than it should be. Shear locking, membrane locking and volume (incompressibility) locking are the most popular locking problems as well as others with no special names [4, p. 299]. Another approach of the term locking is the excessive stiffness in one or more deformation mode [5, p. 95].

## 4.2 Shear locking

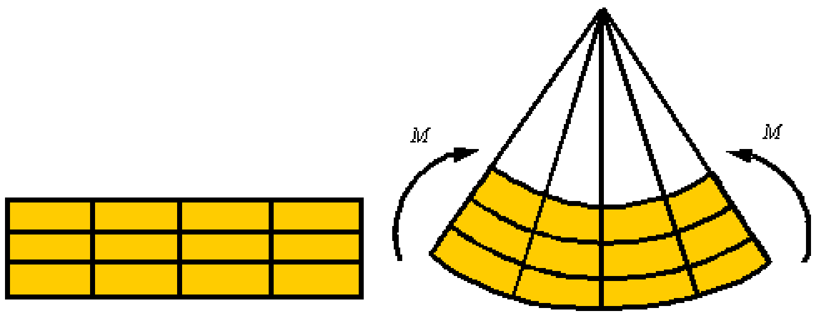


**Fig. 4-1** Pure Bending **[6]**

Before discussing the shear locking problem, we have to review the classical beam theory (It is called Bernoulli’s beam in literature too). In this theory, two important assumptions have been made. First, the contribution of the shear deformation is negligible compared to the rotation, <u>this is equivalent to consider the plane sections of the beam normal to the longitudinal axes will remain plane and perpendicular to the new Neutral Axes</u> while the top and bottom edges of the beam become arcs of the same curvature (as described in the deformed shape of the beam subjected to pure bending in Fig.4.1). The second assumption is that the thickness of the beam should be small compared to the other dimensions [2, p. 49] [3, p. 405].

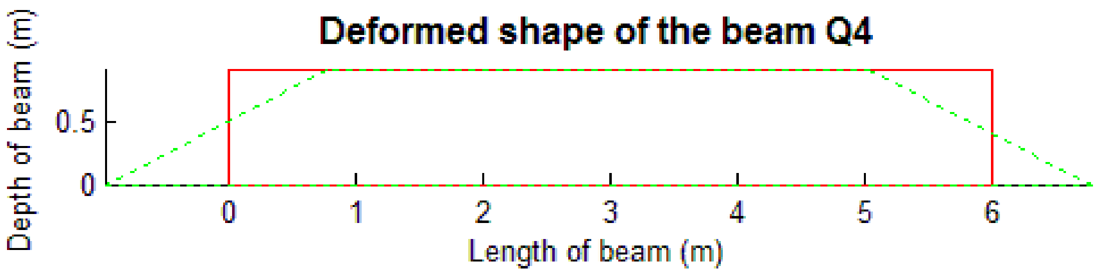


**Fig. 4-2** Deformed shape of a simple beam subjected to pure bending

Fig.4.2 shows a simple beam subjected to pure bending and modeled by Q4 element. Unlike the classical beam theory, the edges of the beam are inclined having an acute angle with the Neutral axis as a result of the displacement field of Q4 element in Eq.3.7. This acute angle is a sign of development of shear strains (strain field in Eq.3.8) and consequently shear stresses. The developed shear stresses lead to over-stiffness in the beam element; that is, the bending moment is resisted by both flexure and shearing stresses. Qualitative results for this problem appear in the analysis of Q4 element; in Fig.3.15a&b, both the deformed shape of the beam and the normal stress of FE model are less than the exact values while the shear stress in Fig.3.15c is larger than its exact value.

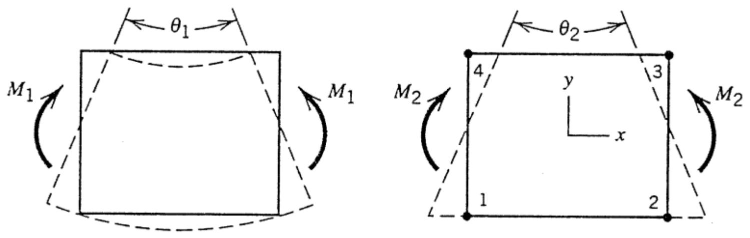


**Fig. 4-3** Pure bending and bending in Q4 element

**[2, p. 52]**

$$M_2 = \frac{1}{1+\nu}\left[\frac{1}{1-\nu} + \frac{1}{2}\left(\frac{a}{b}\right)^2\right] M_1$$

**Eq. 4-1** Relation between pure bending moment and moment due to Q4 element

**[2, p. 52]**

Another approach to define the shear locking could be explained as follow. The two beams in Fig.4.3 have the same properties (Poisson's ratio, elasticity…etc.). Apply moments in each beam ($M_1$ and $M_2$) to get equal angles ($\theta_1$ and $\theta_2$). Eq.4.1 shows the relation between the two bending moments where a/b is the aspect ratio (α is the length of the beam and b is the height). If the aspect ratio increases so much which means insufficient mesh refinement (as shown in Fig.4.2 where the beam under pure bending is modeled by only one element) the moment of the Q4 element will increase so much compared to the exact one, in other meaning, the element becomes infinitely stiff in bending and this phenomena called *locking*. Practically, this large aspect ratio is avoided by good mesh refinement (as shown in the analysis of the Q4 element) and the FE doesn't lock but it becomes stiffer in bending as explained in the previous paragraph [2, p. 52].

Two remedies are discussed for this problem in the following subsections. The first remedy is supplementing the element with additional modes to allow some flexibility for the element. The second remedy is using reduced integration method to integrate the global stiffness matrix **K**. These remedies are applicable only for 4-node elements and cannot be applied on CST elements.

### *4.2.1 Improved Bilinear Quadrilateral Element Q6*

$$u = \sum_{i=1}^{4} N_i u_i + (1-\xi^2)\alpha_1 + (1-\eta^2)\alpha_2$$
$$v = \sum_{i=1}^{4} N_i v_i + (1-\xi^2)\alpha_3 + (1-\eta^2)\alpha_4$$

**Eq. 4-2** Displacement field of Q6 element **[7, p. 47]**

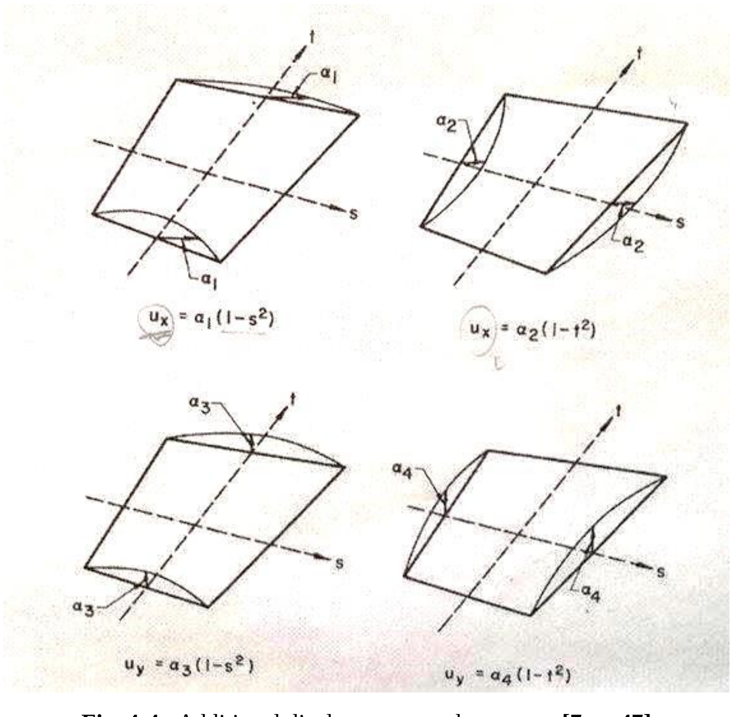


**Fig. 4-4** Additional displacement modes **[7, p. 47]**

Incompatible elements or Improved Q4 element is new element has been developed with six shape functions as shown in Eq.4.2. The two additional displacement modes shown in Fig.4.4 allow some flexibility in the beam and satisfy the pure bending conditions. $\alpha_i$ is the generalized coordinates. They also called nodeless d.o.f. The additional displacement mode accompanied by them are called incompatible displacement field because they allow overlaps or gaps between the elements; each element has its own nodeless d.o.f. and they are not connected to each other.

The strain of a Q6 element in Eq.4.2 consists of two components, the strain of the nodal d.o.f and the nodeless d.o.f. The deformation matrix [**B**] for this element is expanded to include the incompatible displacements; as shown in Eq.4.3, it consists of the compatible displacements of Q4 element [$\mathbf{B_c}$] and the additional incompatible displacements [$\mathbf{B_{ic}}$]. [$\mathbf{B_c}$] is the same as calculated before in Q4 element (Eq.4.5). Jacobian matrix is based only on the nodal d.o.f, so it is not changed. The incompatible displacement field is added to the shape function matrix and its deformation matrix can be found as in Eq.4.6. The stiffness matrix of a Q6 element in Eq.4.11 consists of four matrices illustrated from Eq.4.7-4.10. The displacement mode of Eq.4.12 is shown in Fig.4.5 [5, pp. 219-220] [8].

$$\{\boldsymbol{\varepsilon}\} = [\boldsymbol{B_c}][\boldsymbol{B_I}]\begin{Bmatrix}\boldsymbol{d_c}\\ \boldsymbol{d_I}\end{Bmatrix}$$

**Eq. 4-3** Strain field of a Q6 element

$$[\boldsymbol{B}] = [\boldsymbol{B_c}][\boldsymbol{B_i}]$$

**Eq. 4-4** Deformation matrix of Q6 element

$$[\boldsymbol{B_c}] = \frac{1}{|\boldsymbol{J}|}[\boldsymbol{\Gamma}]\begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & 0 & N_{3,\xi} & 0 & N_{4,\xi} \\ 0 & N_{1,\xi} & 0 & N_{2,\eta} & 0 & N_{3,\eta} & 0 & N_{4,\eta} \end{bmatrix}$$

$$\text{; where } [\boldsymbol{\Gamma}] = \begin{bmatrix} J_{22} & -J_{12} & 0 & 0 \\ 0 & 0 & -J_{21} & J_{11} \\ -J_{21} & J_{11} & J_{22} & -J_{12} \end{bmatrix}$$

**Eq. 4-5** Deformation matrix of the compatible displacement

$$[\boldsymbol{B_I}] = \frac{1}{|\boldsymbol{J}|}[\boldsymbol{\Gamma}]\begin{bmatrix} -2\xi & 0 & 0 & 0 \\ 0 & -2\eta & 0 & 0 \\ 0 & 0 & -2\xi & 0 \\ 0 & 0 & 0 & -2\eta \end{bmatrix}$$

**Eq. 4-6** Deformation matrix of Incompatible displacement

$$K_{CC_{8\times8}} = t\int_{-1}^{1}[\boldsymbol{B_c}]^T_{8\times3}[\boldsymbol{E}]_{3\times3}[\boldsymbol{B_c}]_{3\times8}|\boldsymbol{J}|d\xi d\eta$$

**Eq. 4-7**

$$K_{CI_{8\times4}} = t\int_{-1}^{1}[\boldsymbol{B_c}]^T_{8\times3}[\boldsymbol{E}]_{3\times3}[\boldsymbol{B_I}]_{3\times4}|\boldsymbol{J}|d\xi d\eta$$

**Eq. 4-8**

$$K_{IC_{4\times8}} = t\int_{-1}^{1}[\boldsymbol{B_I}]^T_{4\times3}[\boldsymbol{E}]_{3\times3}[\boldsymbol{B_c}]_{3\times8}|\boldsymbol{J}|d\xi d\eta$$

**Eq. 4-9**

$$K_{II_{4\times4}} = t\int_{-1}^{1} [\boldsymbol{B_I}]^T_{4\times3}[\boldsymbol{E}][\boldsymbol{B_I}]_{3\times4}|\boldsymbol{J}|d\xi d\eta$$ **Eq. 4-10**

$$K_{12\times12} = \begin{bmatrix} K_{CC} & K_{CI} \\ K_{IC} & K_{II} \end{bmatrix}$$ **Eq. 4-11** Stiffness matrix of Q6 element

$$\{\boldsymbol{d_c}\} = [u_1 \quad v_1 \quad u_2 \quad v_2 \quad u_3 \quad v_3 \quad u_4 \quad v_4]^T$$ **Eq. 4-12** Displacements of nodal d.o.f

$$\{\boldsymbol{d_I}\} = [u_5 \quad u_6 \quad v_5 \quad v_6]^T$$ **Eq. 4-13** Displacement of nodeless d.o.f

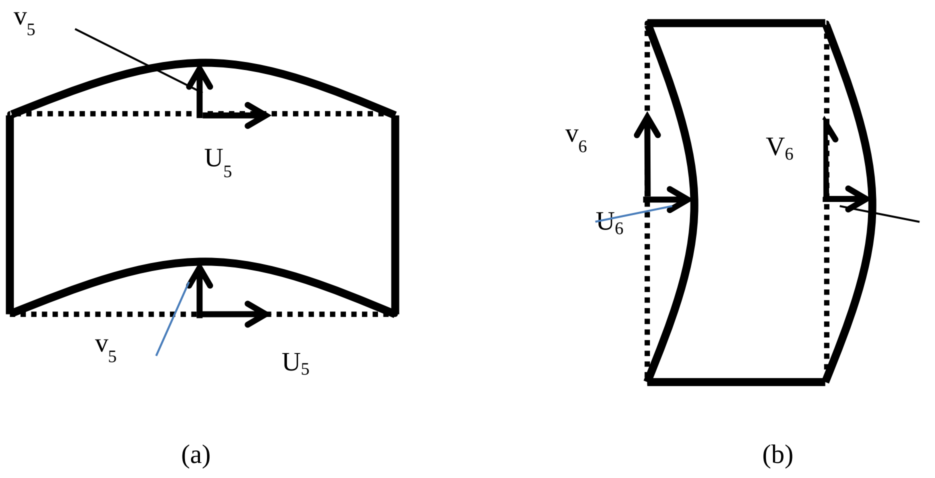


**Fig. 4-5** Displacement modes of Q6 element

## Defects of Q6 element

However, Q6 element in this way failed to represents a state of constant stress unless they are rectangular i.e. consider a mesh of non-rectangular elements is loaded in a way to prevail a state of constant stress, in case of absence of nodeless d.o.f (Q4 element), the elements will behave properly. But in case of the presence of the nodeless d.o.f., the elements will not respond properly. The nodeless d.o.f. should be zeroes and they are not.

A remedy for this defect can be done by adding initial stresses vector $\{\sigma_0\}$ to represent the state of uniform stresses. We seek for zeros nodeless d.o.f. when applying uniform stresses. So a patch test (Eq.4.14) is required. To satisfy this patch test, deformation matrix of the incompatible d.o.f. should be modified by adding a corrected deformation matrix as shown in Eq.4.15. This corrected deformation matrix $[\mathbf{B}_{\mathbf{i\ corr}}]$ (Eq.4.19) is solved by substituting Eq.4.15 in Eq.4.14 [5, p. 220] [8, pp. 6-4].

$$\int [\boldsymbol{B_i}]^T\{\boldsymbol{\sigma_0}\}\, dV = \{0\}, \qquad \text{hence} \qquad \int [\boldsymbol{B_i}]^T\, dV = \{0\}$$ **Eq. 4-14** Patch test condition

$$[\boldsymbol{B_{i\,new}}] = [\boldsymbol{B_i}] + [\boldsymbol{B_{i\,corr}}]$$

**Eq. 4-15** Modified deformation matrix

$$\int [\boldsymbol{B_{i\,new}}]^T \, dV = \{0\}$$

**Eq. 4-16**

$$\int ([\boldsymbol{B_i}] + [\boldsymbol{B_{i\,corr}}]) \, dV = \{0\}$$

**Eq. 4-17**

$$\int [\boldsymbol{B_i}] \, dV + V[\boldsymbol{B_{i\,corr}}] = \{0\}$$

**Eq. 4-18**

$$[\boldsymbol{B_{i\,corr}}] = \frac{-1}{V} \int [\boldsymbol{B}_i] dV$$

**Eq. 4-19** Corrected Deformation matrix

Another defect of Q6 element is incompatibility between elements. The nodeless d.o.f. ($\alpha_1$ to $\alpha_4$) are not connected between the elements rather than the nodal d.o.f. This may allow some gaps or overlaps between the elements as shown in Fig.4.6. However, this defect could be avoided by good mesh refinement, thus incompatibility can be neglected [2, p. 54].

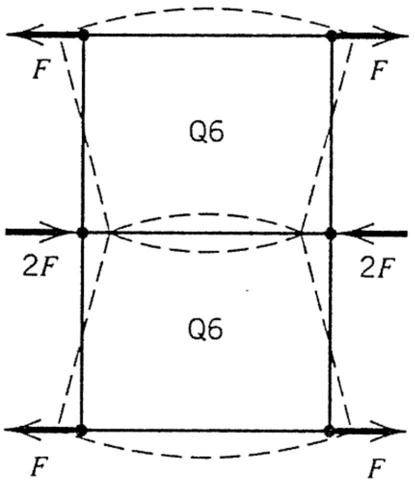
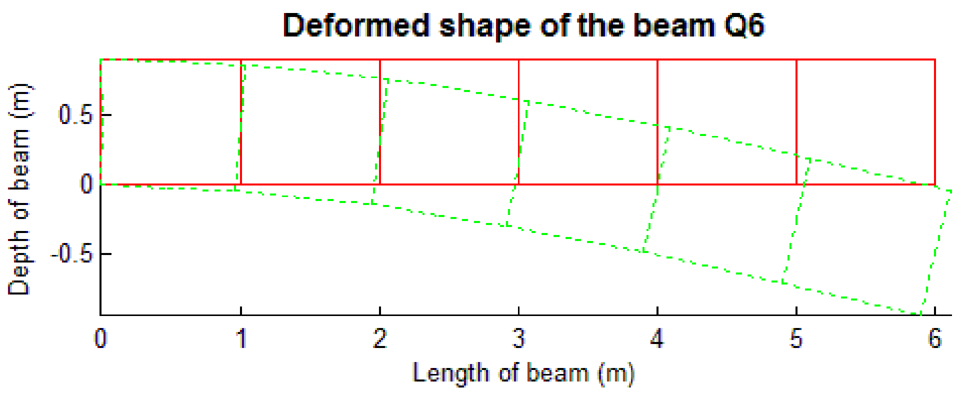


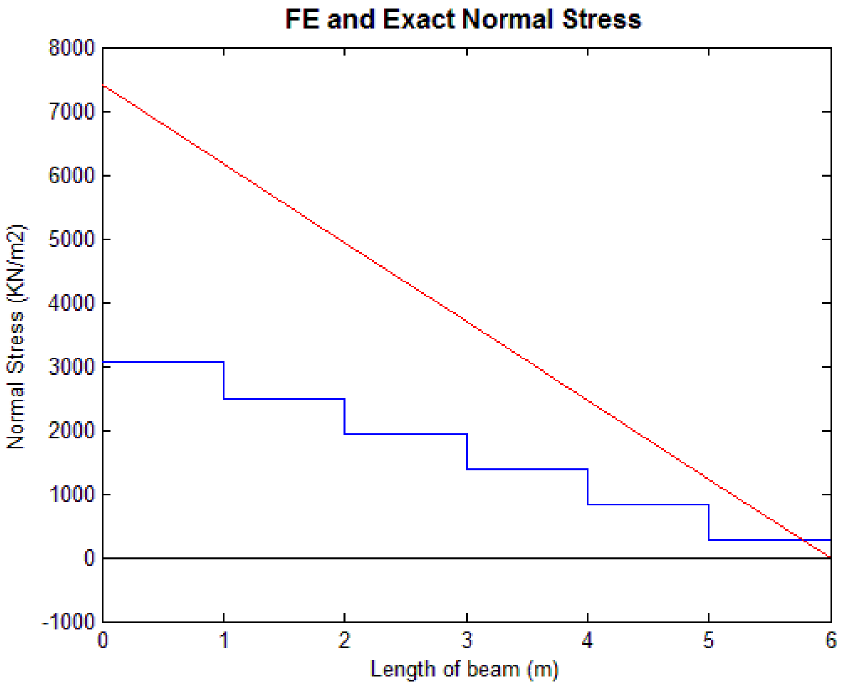
**Fig. 4-6** Incompatibility modes in Q6 element **[2, p. 53]**

**Illustrative example**

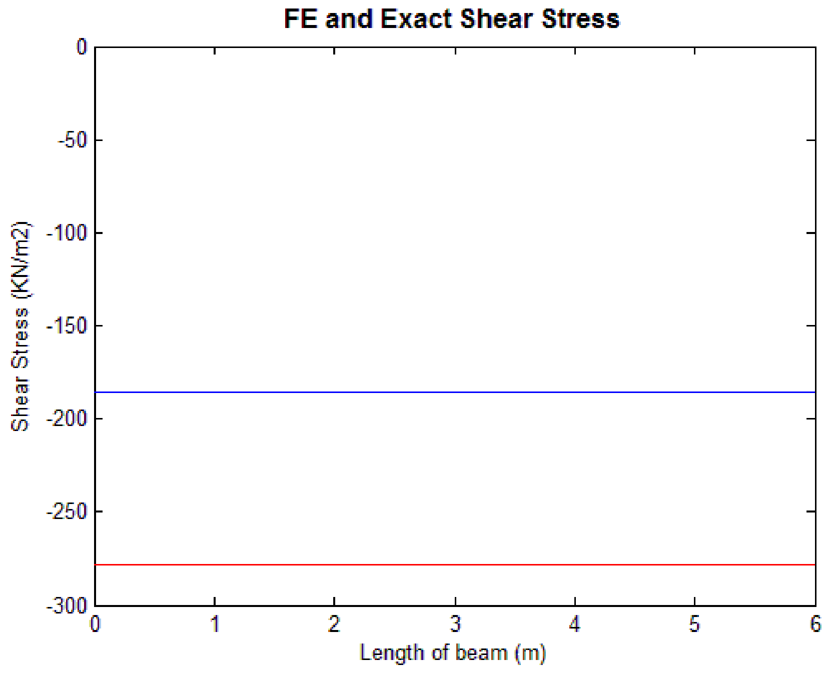
The cantilever beam in the previous examples is now modeled by the Q6 element. The material and cross section properties are the same as used before.



(a) v = $-9.36*10^{-4}$ m

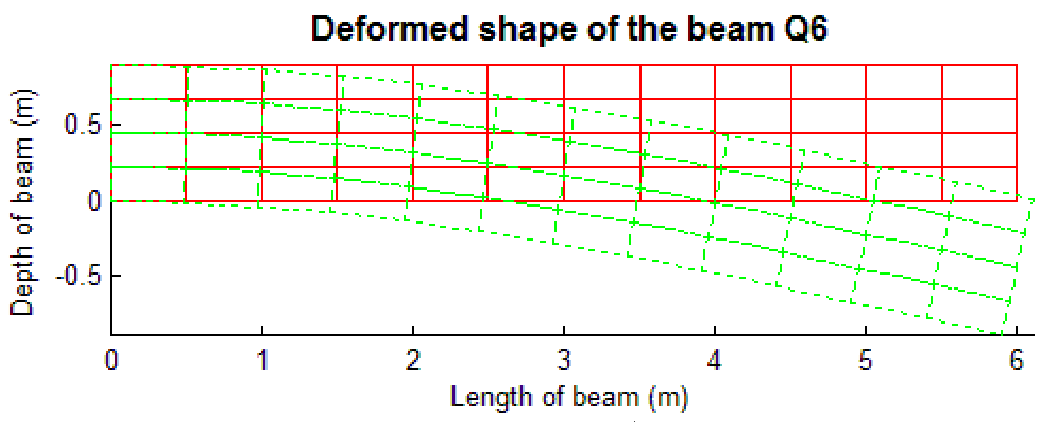


(b) $\sigma_x = 3056$ KN/m$^2$

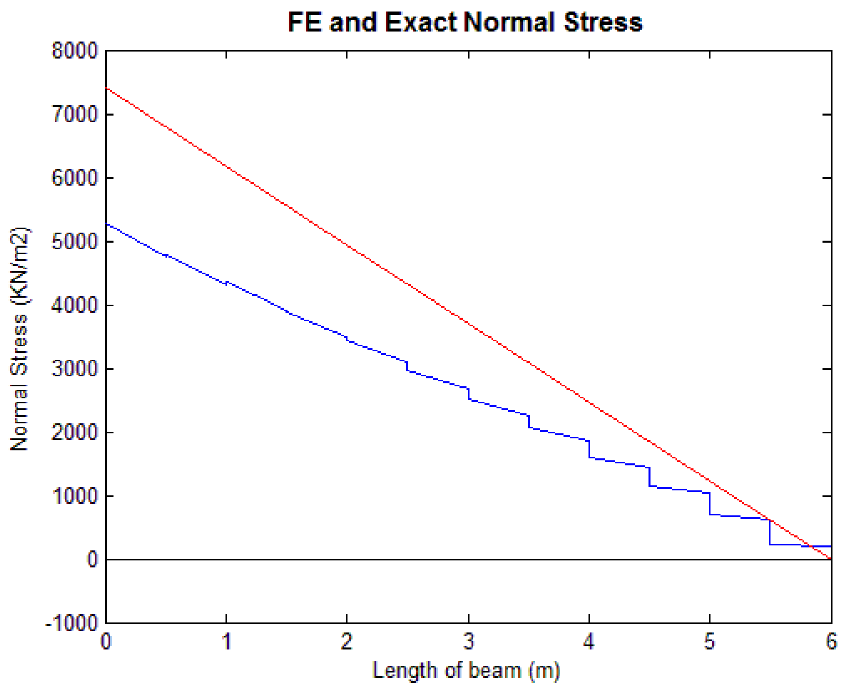


(c)

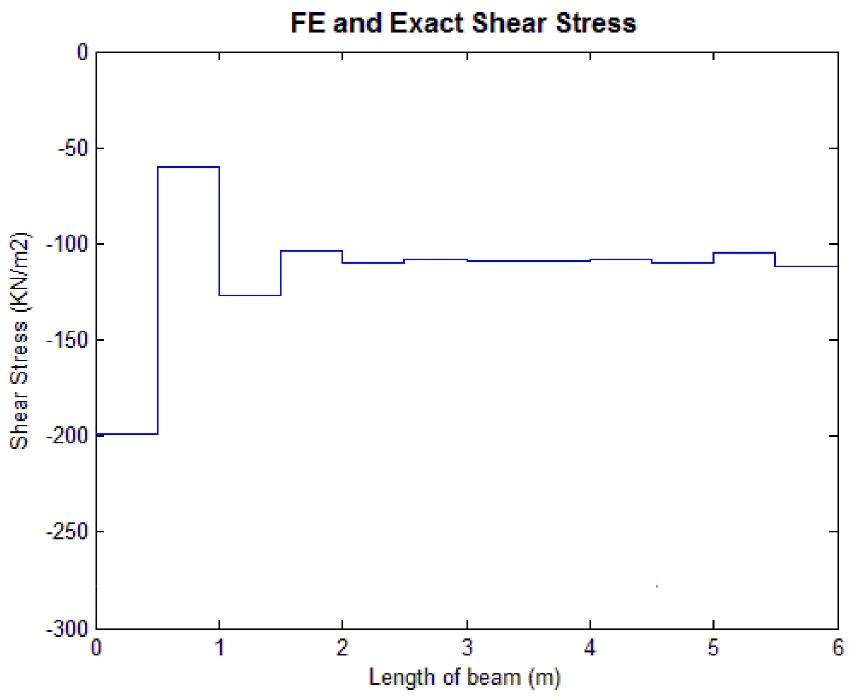
**Fig. 4-7** Analysis of Cantilever_C2_01_03 modeled by Q6 element



(a) v = $-8.9*10^{-4}$ m

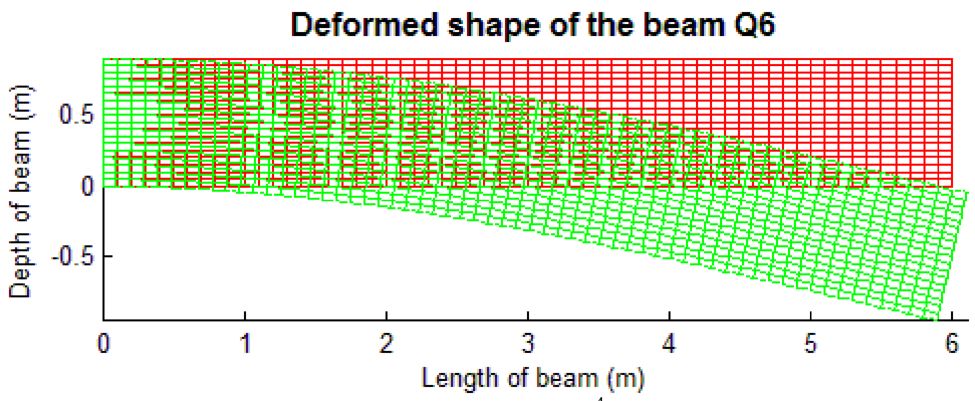


(b) $\sigma_x$ = 5848 KN/m$^2$

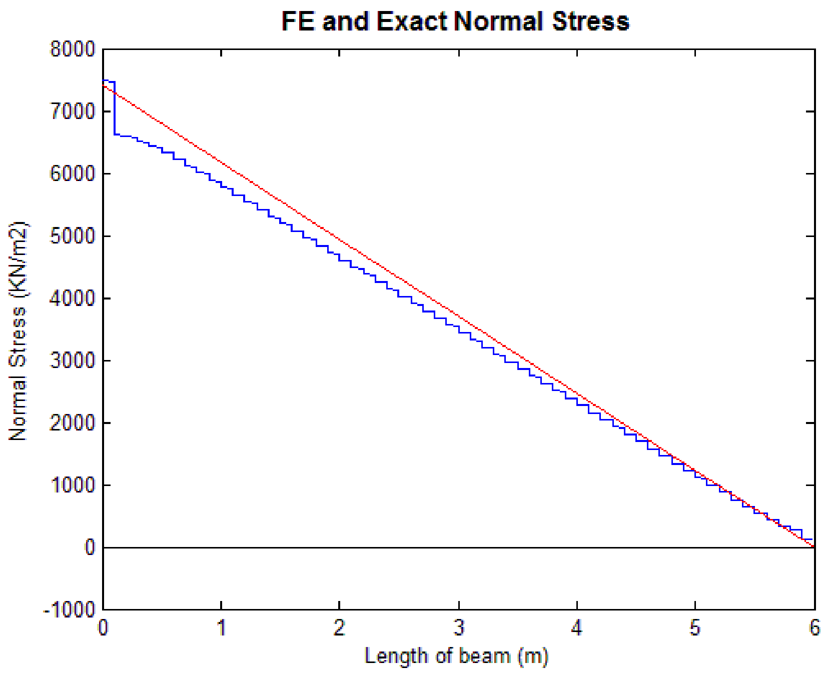


(c)

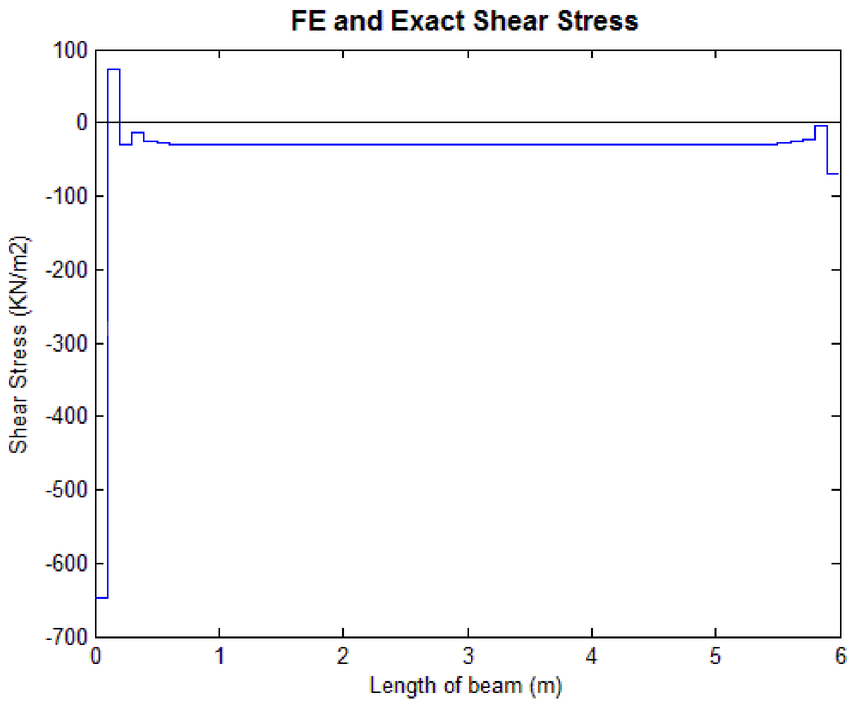
**Fig. 4-8** Analysis of Cantilever_C2_03_03 modeled by Q6



(a) $v = -9.5*10^{-4}$ m



(b) $\sigma_x = 7495$ KN/m$^2$



(c)

**Fig. 4-9** Analysis of Cantilever_C2_05_03 modeled by Q6

| | | Exact | Q4 element | Q6 element |
|---|---|---|---|---|
| **Normal stress ($KN/m^2$)** | 6 elements | 7407 | 2133 | 3056 |
| | 60 elements | | 4830 | 5848 |
| | 1080 elements | | 7424 | 7495 |
| **Deflection (m)** | 6 elements | $-9.406*10^{-4}$ | $-6.073*10^{-4}$ | $-9.36*10^{-4}$ |
| | 60 elements | | $-8.449*10^{-4}$ | $-8.9*10^{-4}$ |
| | 1080 elements | | $-9.49*10^{-4}$ | $-9.5*10^{-4}$ |

**Table 4-1** Results of the analysis of the beam in different elements

From Fig.4.7 to 4.9, we notice that the Q6 element gives good results in normal and shear stresses[12]. The deformed shape of the beam is flexible. Table 4.1 shows the results that Q6 element improves the results of the normal stress; the normal stress in Q6 element is approaching the exact value rather than that of Q4 element. We get rid of the shear locking problem and now we have higher normal stresses and deflection than previous. Also, when you compare Fig.4.7c to 4.9c to their corresponding –Fig.3.16c to Fig.3.18c- a reduction in the shear stresses happened because the beam now in Q6 element is no longer has over-stiffness in bending.

Now, we can summarize that Q6 element has succeeded to overcome the shear locking problem and reduce the over-stiffness of the beam in bending. The FE results are approaching the exact values with less mesh refinement used.

### *4.2.2 Weighted Integration method*

Comparing the exact strain energy of the element to that modeled by FEA is a way to know whether the element is over-stiffened or not. The energy ratio r in Eq.4.20 shows the relation between the strain energy of 4-node element $U_{quad}$ and the exact one of the beam $U_{beam}$. If it is greater than 1, then the element is over-stiffened. If it is less than 1, it is under-stiffened, and the result is exact if it is equal to 1 [1, pp. 17-16].

$$r = \frac{U_{quad}}{U_{beam}}$$

**Eq. 4-20** Energy ratio

[12] The MatLab code used for this model explained in Appendix E (Section 3.5)

$$U_{quad} = \frac{1}{2}[\boldsymbol{u_{beam}}]^T[\boldsymbol{K^e}]\{\boldsymbol{u_{beam}}\}$$

**Eq. 4-21** Strain energy of 4-node element

$$U_{beam} = \frac{1}{2}M\kappa a = \frac{M^2 a}{2EI} = \frac{6M^2}{Eha^2\gamma^3}$$

**Eq. 4-22** Exact strain energy of the corresponding beam segment

$$r = \frac{1 - \frac{2}{\gamma^2} - v}{\left(\frac{2}{\gamma^2}\right)(1 - v^2)}$$

**Eq. 4-23** Energy ratio **[9, pp. 17-17]**

We are seeking for the value of r in terms of the dimension of the element. Consider a beam subjected to pure bending and is divided to 4-node elements each of length α and height b. We are comparing the strain energy of the 4-node element $U_{quad}$ to the exact strain energy from mechanics of materials. The strain energy in 4-node elements is computed using Eq.4.21, where $\mathbf{u_{beam}}$ is the nodal displacement vector and consists of $\mathbf{u_x}$ and $\mathbf{u_y}$, and $\mathbf{K^e}$ is the stiffness matrix of the element. But $\mathbf{K^e}\times\mathbf{u_y}$ vanishes because this is a state of pure bending and no shear forces are developed. The corresponding strain energy in this beam segment is calculated using Eq.4.22, where M is the applied bending moment and κ is the curvature of the beam. By substituting both of Eq.4.21 and Eq.4.22 in Eq.4.20, we get the energy ratio r in terms of the aspect ratio γ (b/a) and Poisson's ratio ν.

From Eq.4.23, we can notice that if γ is much larger than 1, r will be much smaller than 1 and the element is under-stiffened. While if γ is much smaller than 1, r will be much larger than 1 and the element is over-stiffened. This is similar to what we had discussed before in the beginning of this section, specifically in Eq.4.1. High values of r cause shear locking problems.

A remedy for this problem is to compensate the difference in the energy ratio by getting an adjusted stiffness matrix using the weighted integration method. It is better to get the stiffness matrix of an element by using a linear combination of stiffness produced by two different integration rules and that is called *weighted integration method*[13]. Eq.4.24 shows the stiffness matrix of 4-node element done by combining two stiffness matrices integrated by 1×1 and 2×2 gauss points, while β is a factor to reduce the shear locking effect (Eq.4.25&4.26). In the linear combination of Eq.4.24, we can

[13] More discussion about using of numerical integration method can be found in chapter 5

notice that K $_{1\times1}$ is too soft and K$_{2\times2}$ is too stiff. Such a combination may give balanced stiffness.

$$K_{\beta}^{e} = (1-\beta)K_{1\times1}^{e} + \beta K_{2\times2}^{e}$$

**Eq. 4-24** Stiffness matrix by weighted integration method

$$r = \frac{\beta\left(1-\frac{2}{\gamma^2}-v\right)}{\left(\frac{2}{\gamma^2}\right)(1-v^2)}$$

**Eq. 4-25**

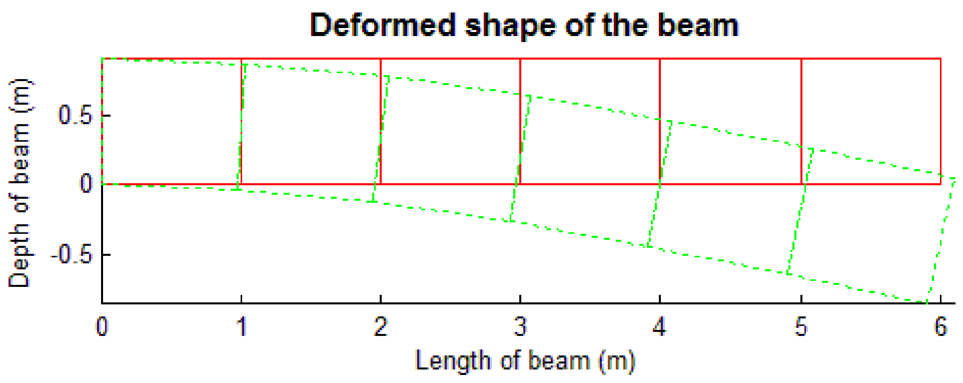
$$\beta = \frac{\left(\frac{2}{\gamma^2}\right)(1-v^2)}{\left(1-\frac{2}{\gamma^2}-v\right)}$$

**Eq. 4-26** Factor to adjust shear locking

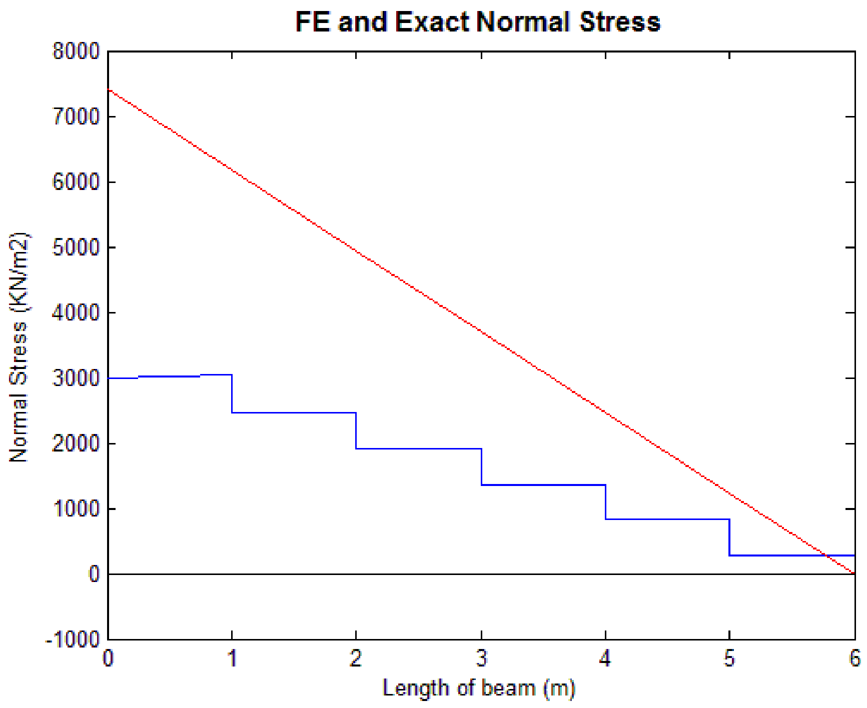
**Illustrative Example**

The same cantilever used before is now modeled using the weighted integration method. We can notice better results compared to Q4 elements as shown in Fig.4.10-4.12. The normal stresses in this case are higher than in Q4 elements and the displacement as well, while the shear stresses are lower as a result of some reduction in the shear locking problem[14].

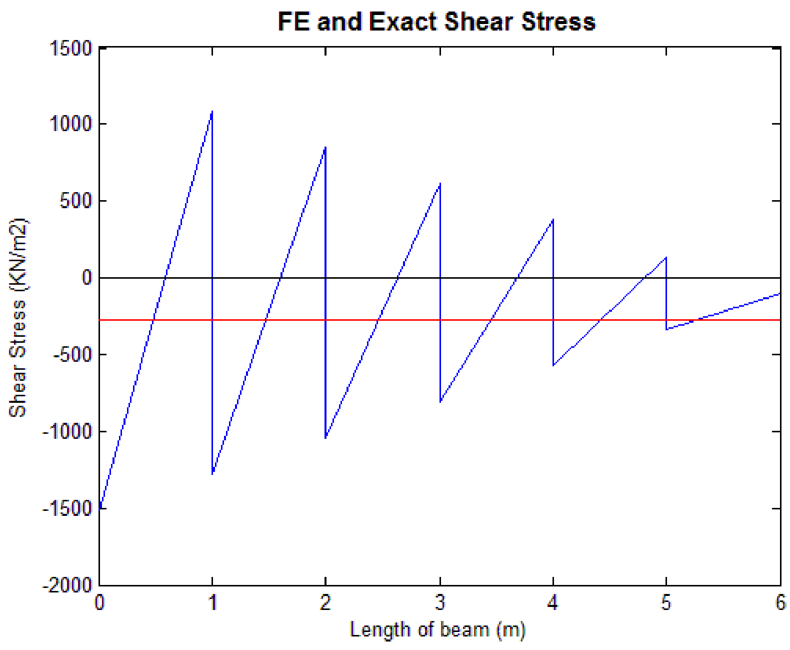
[14] The MatLab code used for this model explained in Appendix F (Section 3.6)



(a) v = -8.54*$10^{-4}$ m

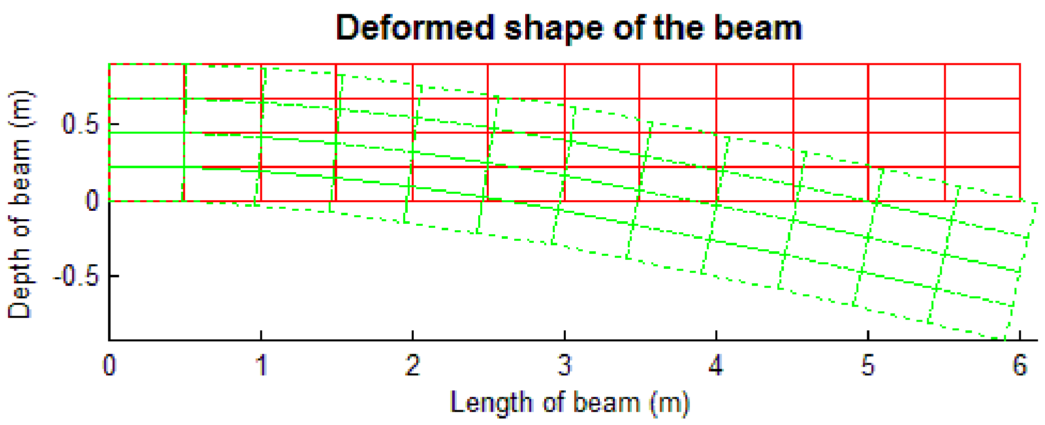


(b) $\sigma_x$ = 2977 KN/$m^2$

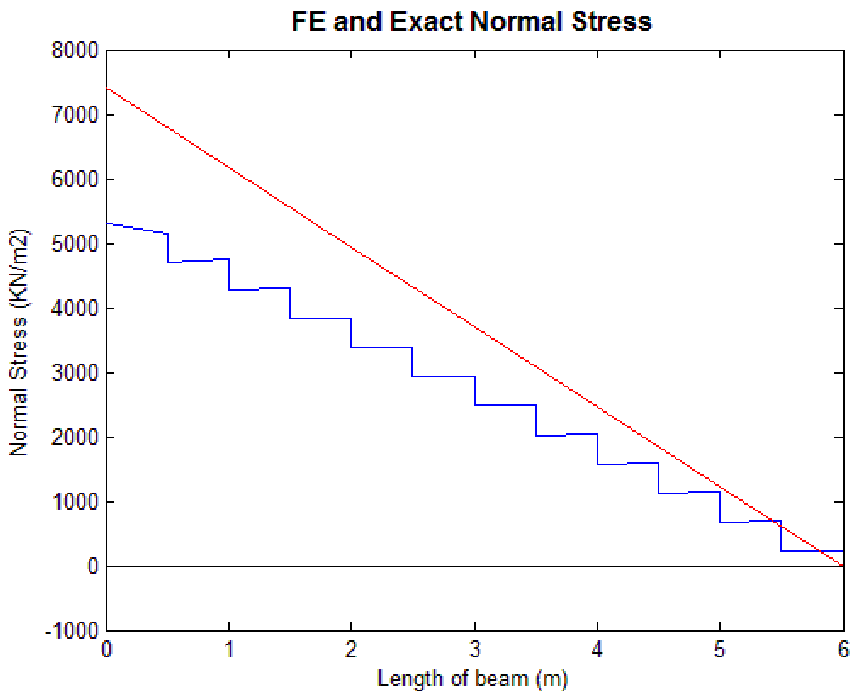


(c)

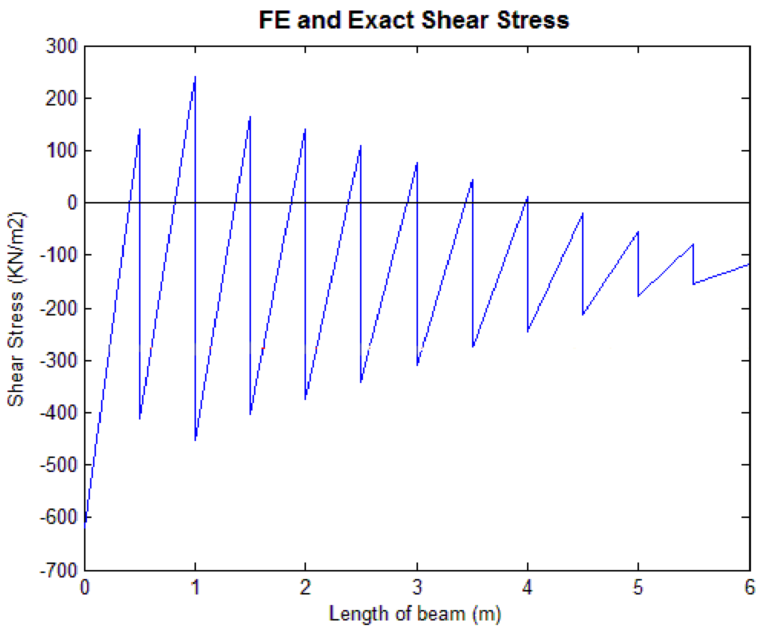
**Fig. 4-10** Analysis of Cantilever_C2_01_03 modeled by Weighted Integration method



(a) $v = -9.26*10^{-4}$ m

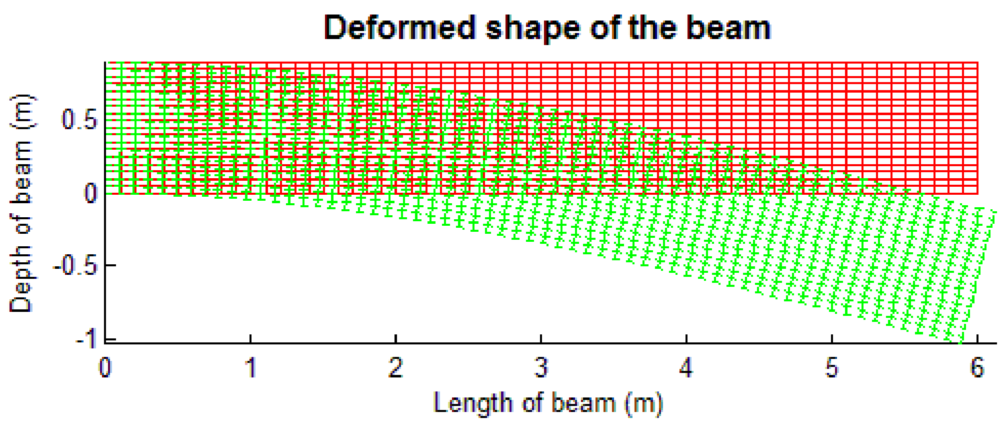


(b) $\sigma_x = 5294$ KN/m$^2$

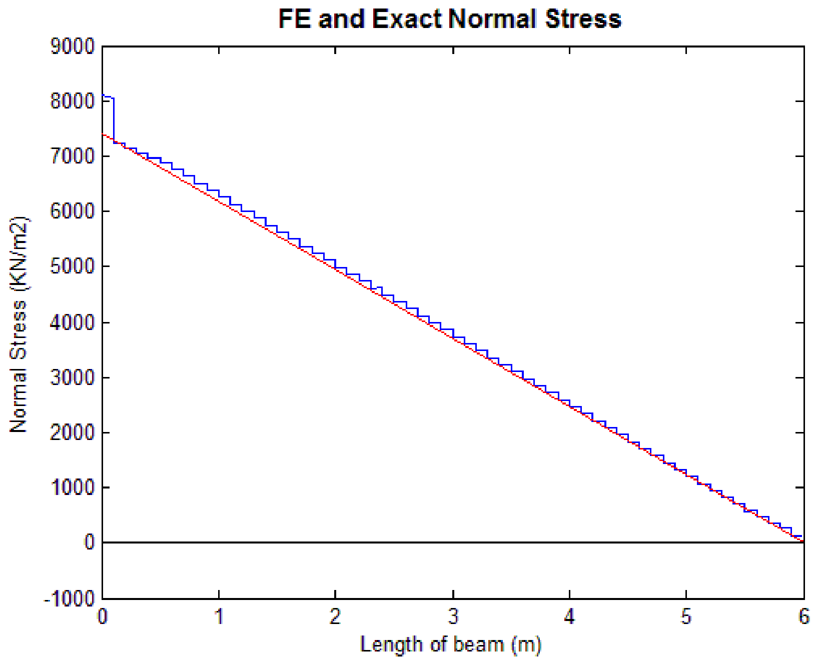


(c)

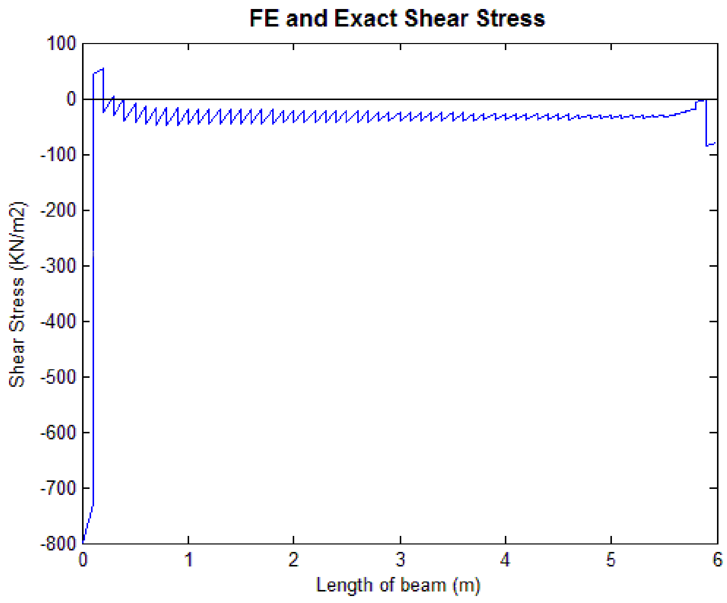
**Fig. 4-11** Analysis of Cantilever_C2_03_03 modeled by Weighted Integration Method



(a) $v = -9.5*10^{-4}$ m



(b) $\sigma_x = 7495$ KN/m$^2$



(c)

**Fig. 4-12** Analysis of Cantilever_C2_05_03 modeled by weighted integration

### *4.2.3 Selective Integration Method*

$$E = E_I + E_{II}$$

**Eq. 4-27**

$$[E_I] = \frac{E}{1-v^2}\begin{bmatrix} \alpha & \beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}$$

$$[E_{II}] = \frac{E}{1-v^2}\begin{bmatrix} 1-\alpha & v-\beta & 0 \\ v-\beta & 1-\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

**Eq. 4-28** Elasticity matrix by Selective Integration Method

$$K_s{}^e = \int_{V^e} t[B]^T[E_I][B]dV^e + \int_{V^e} t[B]^T[E_{II}][B]dV^e = K_I^e + K_{II}^e$$

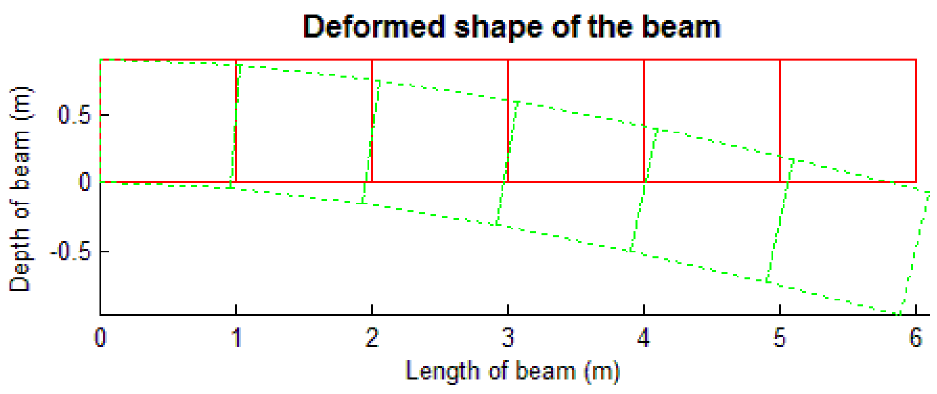
**Eq. 4-29** Stiffness matrix of an element done by Selective Integration Method

Selective Integration Method is more effective than Weighted Integration Method explained in the previous subsection. The Elasticity matrix is divided into two components as what is called stress strain splitting to form $\mathbf{E_I}$ and $\mathbf{E_{II}}$ (Eq.4.27&4.28) where α and β are scalars; ($\alpha = v^2$) and β is the factor to adjust shear locking (Eq.42). The Global Stiffness Matrix $\mathbf{K^e}$ in Eq.4.29 is formulated as the sum of two matrices computed with different integration rules (mainly one part is integrated by reduced integration and the other part is integrated by full integration). $\mathbf{E_I}$ is substituted in the reduced integration part while $\mathbf{E_{II}}$ is substituted in the full integration part. [1, pp. 17-12,17&18].

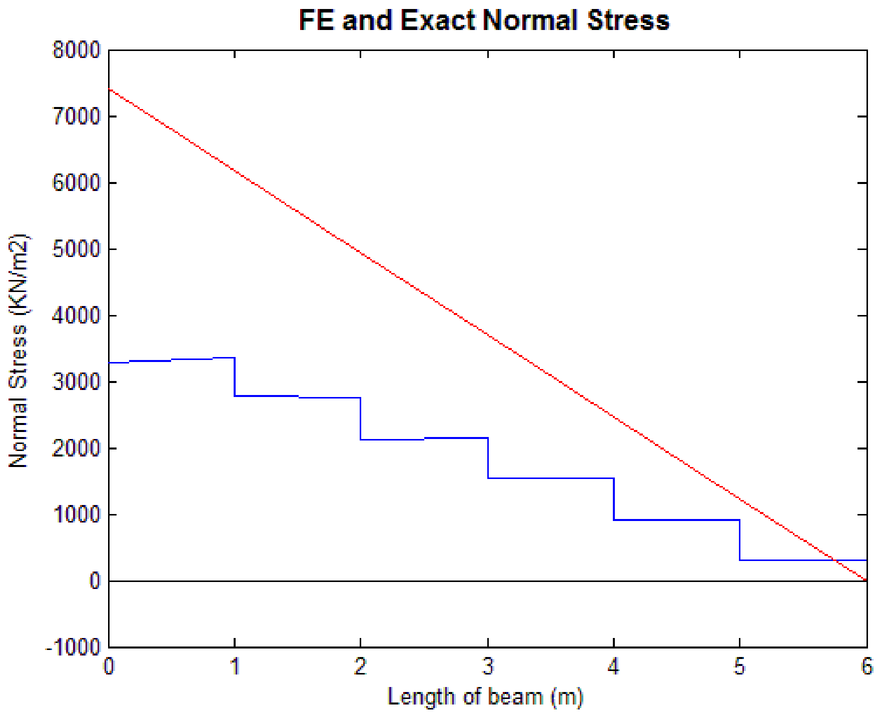
**Illustrative Example**

The previous cantilever is now modeled using the selective integration method[15]:

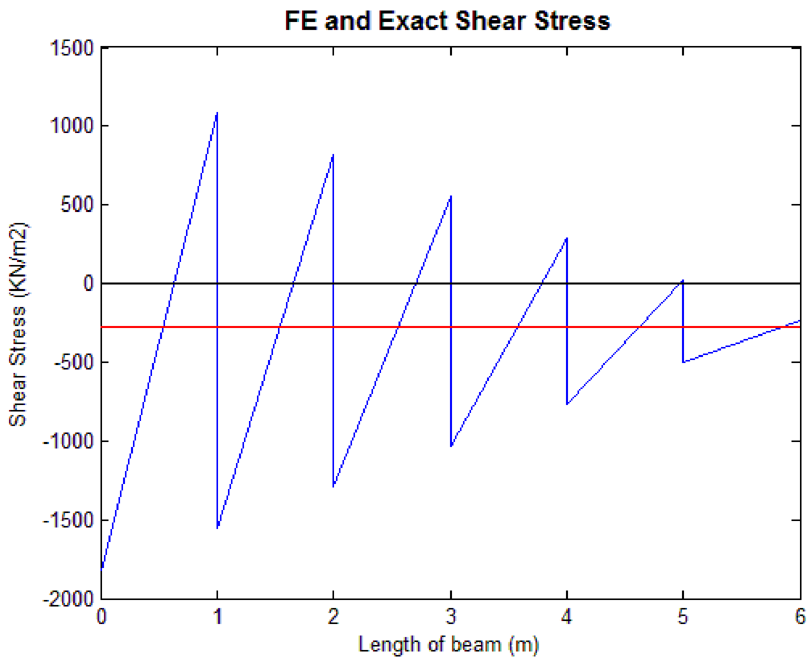
[15] The MatLab code used for this model explained in Appendix G (Section 3.7)



(a) v = $-9.62*10^{-4}$ m

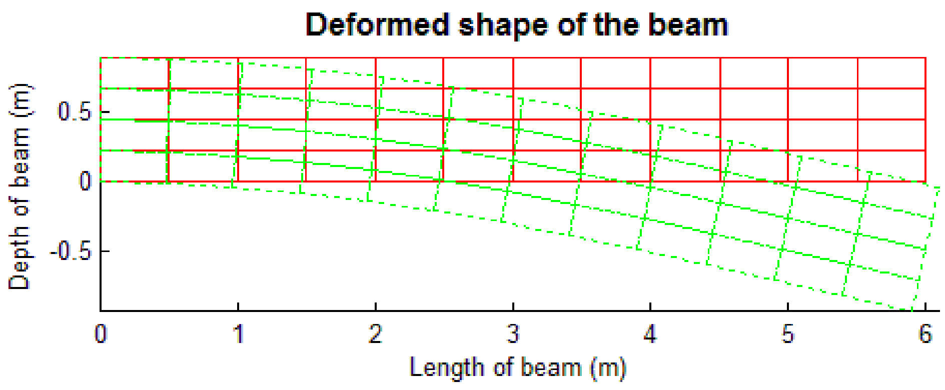


(b) $\sigma_x$ = 3288 KN/$m^2$

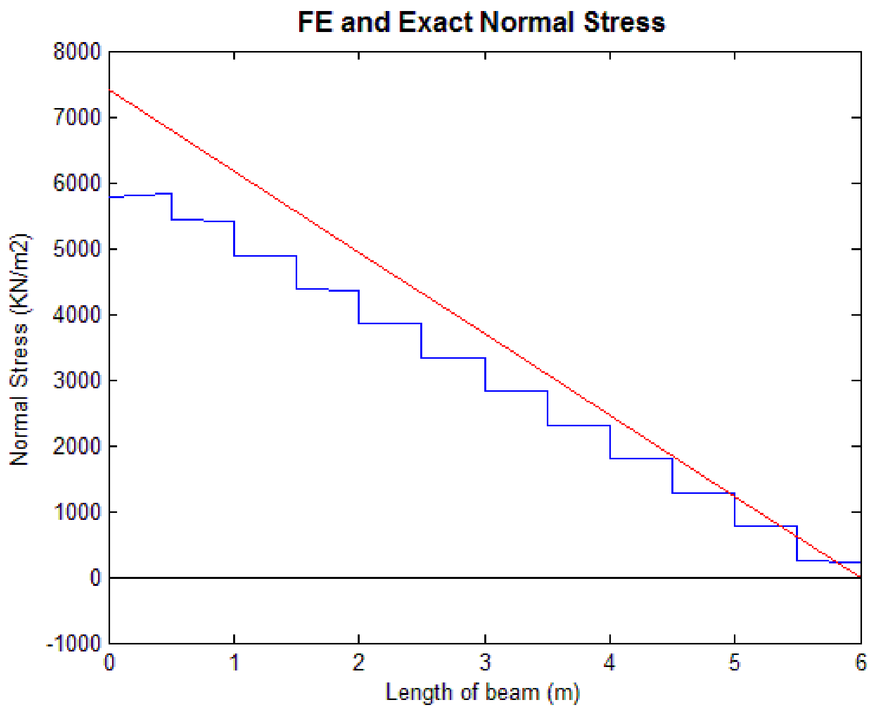


(c)

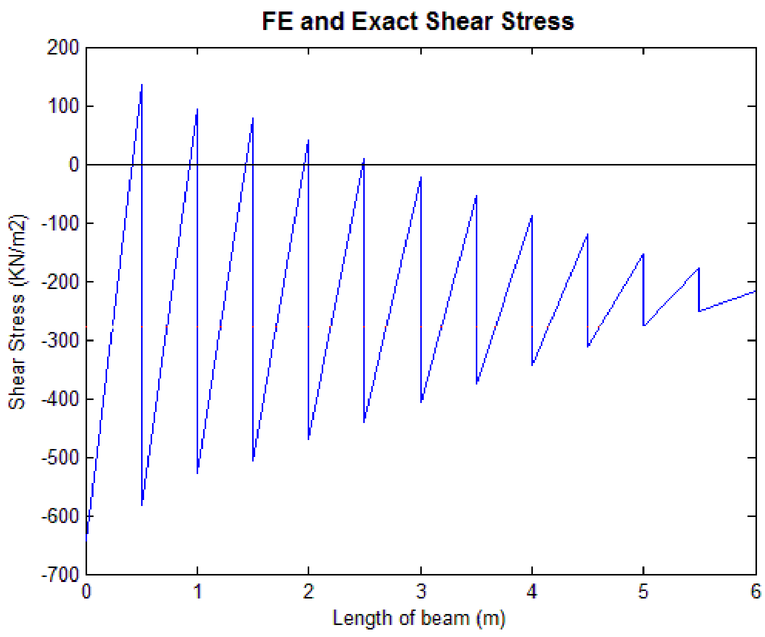
**Fig. 4-13** Analysis of Cantilever_C2_01_03 modeled by Selective Integration method



(a) v = $-9.425*10^{-4}$ m

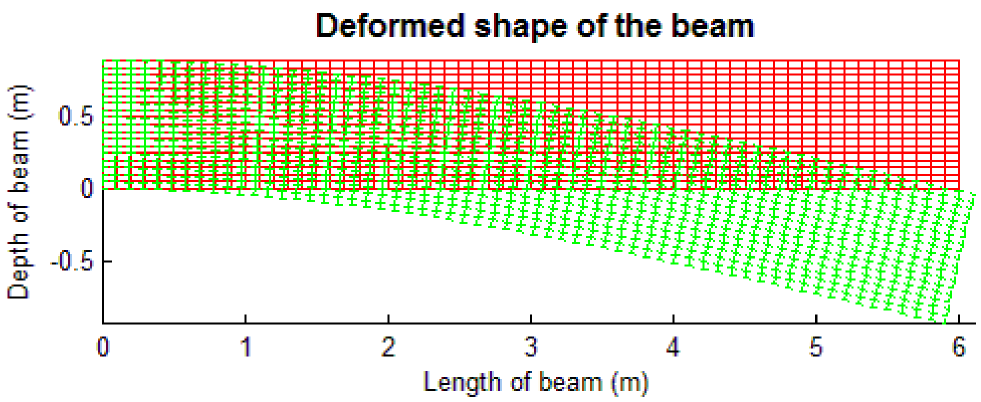


(b) $\sigma_x = 5764$ KN/m$^2$

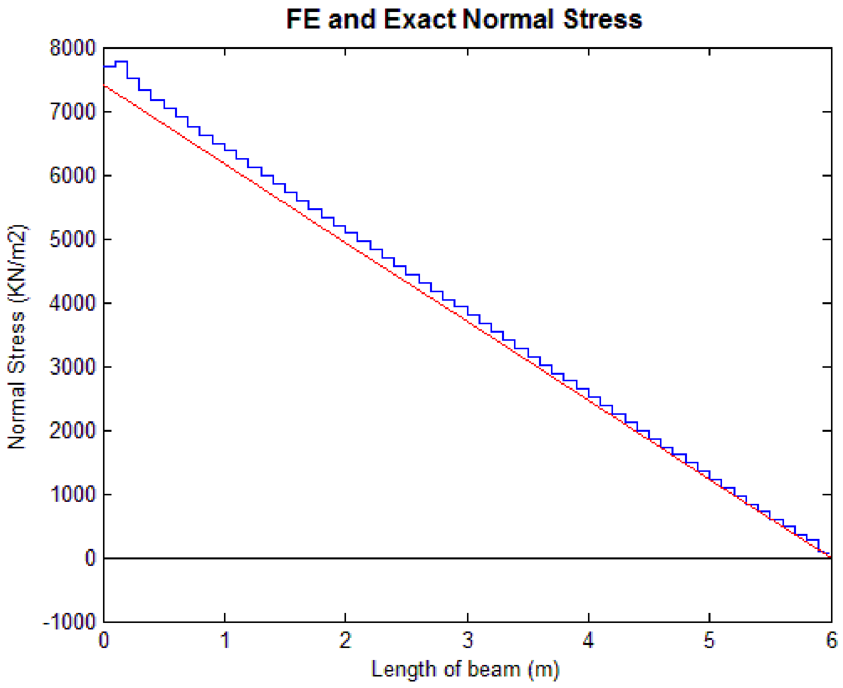


(c)

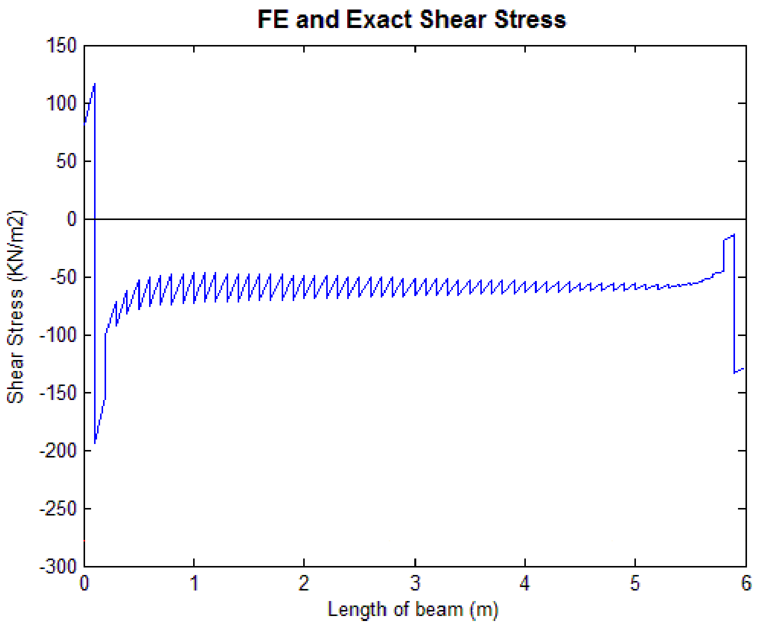
**Fig. 4-14** Analysis of Cantilever_C2_03_03 modeled by Selective Integration Method



(a) $v = -9.5*10^{-4}$ m



(b) $\sigma_x = 7495$ KN/m$^2$



(c)

**Fig. 4-15** Analysis of Cantilever_C2_05_03 modeled by Selective integration

| | | Exact | Q4 element | Weighted Integration | Selective Integration |
|---|---|---|---|---|---|
| Normal stress (KN/m$^2$) | 6 elements | **7407** | 2133 | 2977 | 3288 |
| | 60 elements | | 4830 | 5294 | 5764 |
| | 1080 elements | | 7424 | 7495 | 7780 |
| Deflection (m) | 6 elements | **$-9.406*10^{-4}$** | $-6.073*10^{-4}$ | $-8.54*10^{-4}$ | $-9.62*10^{-4}$ |
| | 60 elements | | $-8.449*10^{-4}$ | $-9.26*10^{-4}$ | $-9.425*10^{-4}$ |
| | 1080 elements | | $-9.49*10^{-4}$ | $-9.5*10^{-4}$ | $-9.43*10^{-4}$ |

**Table 4-2** Analysis of beam using different methods of integration

Table 6 show improved results to their corresponding in both weighted integration method and Q4 element. The shear locking problem is reduced more. Normal stresses is approaching the exact result with less gap and the displacement as well, while the shear stresses are getting lower than its corresponding in both Weighted Integration Method and the Q4 element. A good distribution of stresses and displacement along the beam is shown as well as shown in Fig.4.13-4.15.

## 4.3 Incompressibility locking

Modeling of incompressible objects like fluids (e.g. water) or rubberlike materials causes problems in calculating the elasticity matrix. It is known that the value of Poisson’s ratio (ν) for these materials is equal to 0.5. By substituting this value in Eq.4.30, the elasticity matrix for plane strain problems will tend it to infinity; which means that the pressure required to cause volumetric strain approaches infinity.

To find a solution for this locking problem, we have to understand the physical behavior of the incompressible material. Any material exposed to stresses in two different forms; volumetric or hydrostatic stresses which affects the volume of the material and deviatoric stresses which affects the shape of the material as shown in Eq.4.31&4.32 [10]. So, the fourth-order stiffness tensor $^4C$ can be expressed in terms of volumetric and deviatoric parts as in Eq.4.33 [11, p. 4&6] [12, pp. 85-86].

$$[\boldsymbol{E}] = \frac{E}{(1+v)(1-2v)}\begin{bmatrix} 1-v & v & 0 \\ v & 1-v & 0 \\ 0 & 0 & \frac{1-2v}{2} \end{bmatrix}$$

**Eq. 4-30** 2D Elasticity matrix in plane strain condition

$$[\boldsymbol{\sigma}] = [\boldsymbol{\sigma_{Hyd}}] + [\boldsymbol{\sigma'}]$$

**Eq. 4-31** Hydrostatic & Deviatoric stresses

$$[\boldsymbol{\sigma_{Hyd}}] = \frac{1}{3}tr(\sigma) = \begin{bmatrix} \sigma_{11} & 0 & 0 \\ 0 & \sigma_{22} & 0 \\ 0 & 0 & \sigma_{33} \end{bmatrix}$$

**Eq. 4-32** Hydrostatic stress

$$[\boldsymbol{\sigma'}] = [\boldsymbol{\sigma}] - [\boldsymbol{\sigma_{Hyd}}]$$

**Eq. 4-33** Deviatroric stress

$${}^4C = {}^4C^v + {}^4C^d \text{ where } \sigma = {}^4C : \varepsilon$$

**Eq. 4-34** Fourth-order stiffness tensor

$${}^4C^v = \kappa II = \frac{Ev}{(1+v)(1-2v)}\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Eq. 4-35** Volumetric part of fourth-order stiffness tensor

$${}^4C^d = 2G\left({}^4I^s - \frac{1}{3}II\right) = \frac{E}{(1+v)}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

**Eq. 4-36** Deviatoric part of fourth-order stiffness tensor

The stiffness matrix can be divided into two components; one has to do with the volumetric change of the element and the other with the deviatoric change in the element as shown in Eq.4.37. This formulation of stiffness matrix is a mathematical/weak formulation.

$$K = K^v + K^d = \int_V \nabla\text{N}.\ {}^4C^V.\nabla N^T dV + \int_V \nabla\text{N}.\ {}^4C^d.\nabla N^T dV$$

**Eq. 4-37** Stiffness matrix in the weak formulation

Another formulation for the stiffness matrix can be expressed in terms of the shear and bulk modulus. Bulk modulus expresses the ratio of hydrostatic pressure to the fractional volume change. In incompressible material, the volume change reaches zero and **B** tends to infinity. Thus the incompressible material could be expressed as it has Poisson’s ratio equals to 0.5.

For incompressible materials, the volumetric part in the stiffness matrix governs the equation in both Eq.4.36 and Eq.4.39. It tends to infinity as ν approaches 0.5, thus $\mathbf{k_G}$ becomes singular; in other words it is locked. Solution for this problem is substituting the Poisson's ratio with a value slightly less than 0.5 [5, p. 94 & 497] [11, p. 7].

$$G = \frac{E}{2(1+v)} \qquad \text{and} \qquad B = \frac{E}{3(1-2v)}$$

**Eq. 4-38** Shear and Bulk modulus

$$[E] = G[E_G] + B[E_B]$$

$$[E] = G\begin{bmatrix} \frac{4}{3} & -\frac{2}{3} & -\frac{2}{3} & 0 & 0 & 0 \\ -\frac{2}{3} & \frac{4}{3} & -\frac{2}{3} & 0 & 0 & 0 \\ -\frac{2}{3} & -\frac{2}{3} & -\frac{2}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} + B\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

**Eq. 4-39** Elasticity matrix in terms of shear and Bulk modulus

$$[k] = G\int_V [B]^T[E_G][B]dV + B\int_V [B]^T[E_G][B]dV$$

$$[k] = G[k_G] + B[k_B]$$

**Eq. 4-40** Stiffness matrix in terms of Shear and Bulk modulus

The geometry of the elements may cause volumetric locking, especially for triangular elements. Fig.4-16 shows a plate made of incompressible material and it was divided to triangular elements. This geometric shape caused volumetric locking as follow: Imagine triangle 1 is separated from the plate as shown in Fig.4-17; we are interested in maintaining the area of the triangle to be unchanged. So, if node 1 and 2 in triangle 1 are fixed, there is no way for node 3 to be displaced except in x direction, in other words, node 3 will have $u_3$ only and its $v_3$ is zero. For triangle 2 in Fig.4-17, if node 4 and 5 are fixed, no way for node 6 to be displaced except in y direction only, in other words $u_6$ is zero. If both triangles 1 and 2 combine together as shown in Fig.4-18, node 3 cannot be displaced anymore and the shape is locked [13].

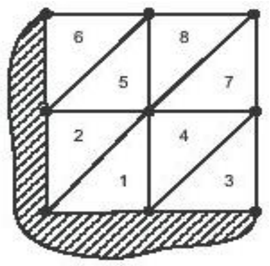


**Fig. 4-16** Triangular meshing exposed to volumetric locking

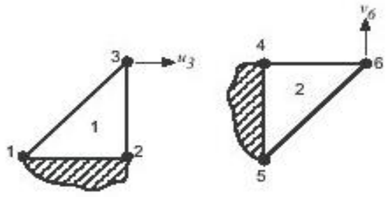


**Fig. 4-17**

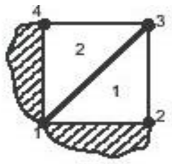


**Fig. 4-18** The triangular meshing is locked

A remedy for this problem is the rearrangement of the nodes to produce another shape able to deform without changing its area/volume. Using crossed triangles as shown in Fig.4-18 will prevent the locking. If nodes A, B and D are fixed, node E can move freely without changing the area of the small triangular elements, thus the whole shape of the plate.

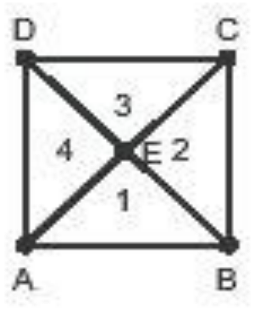


**Fig. 4-19** Crossed triangles

# CHAPTER 5

CHOICE OF NUMERICAL INTEGRATION

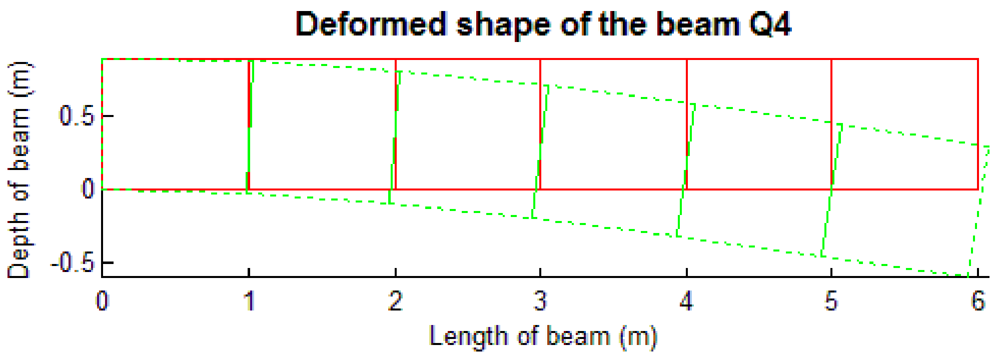
# 5. Choice of Numerical Integration

This chapter discusses how the choice of numerical integration could affect the FEA. Mainly the gauss quadrature method is discussed. A cantilever beam is modeled using different gauss points to show this effect.

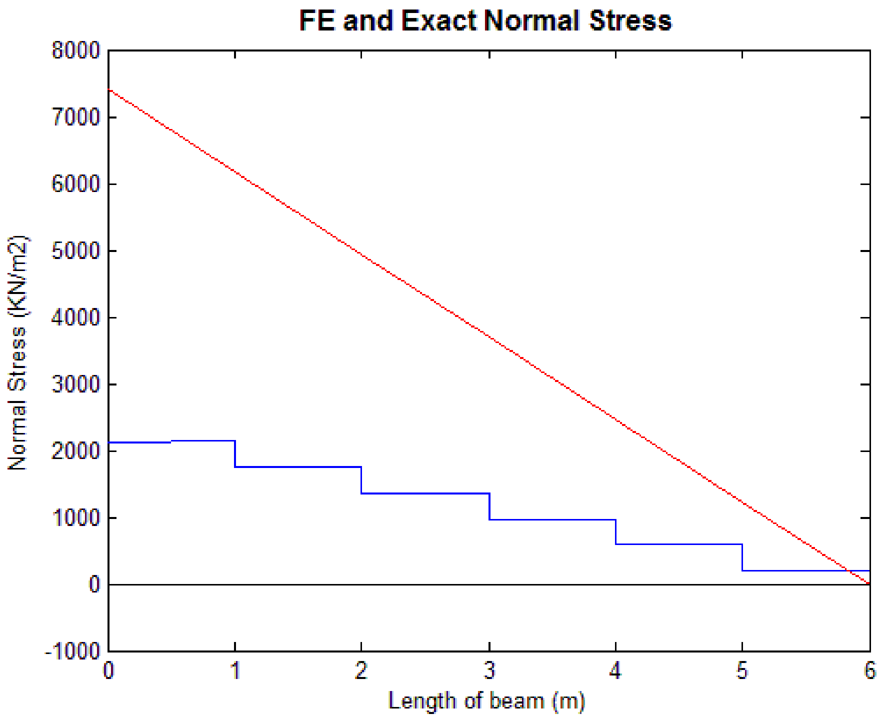
For more generalized shapes with different thickness, analytical integration will be very complicated to solve. In this case, numerical integration of element stiffness is used but it does not provide exact result. Increasing the accuracy of the integration can be done by using more integration points. However, it will not increase the accuracy of the FE analysis. Using more integration (gauss) points leads to over-stiffness in the element because additional points capture more higher-order terms in **k**. These terms resist some deformation modes that lower-order terms do not. On the other hand, using less integration (gauss) points produces worse deformation modes as: instability, spurious singular mode, zero energy mode, hourglass mode, kinematic mode and instability mode. Some ways are being discussed in the following to avoid these spurious modes [5, p. 223] [2, p. 84].

It is known from Gauss Quadrature method if $\phi = \phi(\xi)$ is a polynomial, n-point Gauss quadrature approaches the exact integral if $\phi$ is of degree $2n - 1$ or less. For example, if we have this form$\phi = c_1 + c_2\xi$, one gauss point is enough to get its exact integration and the form$\phi = c_1 + c_2\xi + c_3\xi^2$, two gauss points are sufficient to get its exact integration and so on.

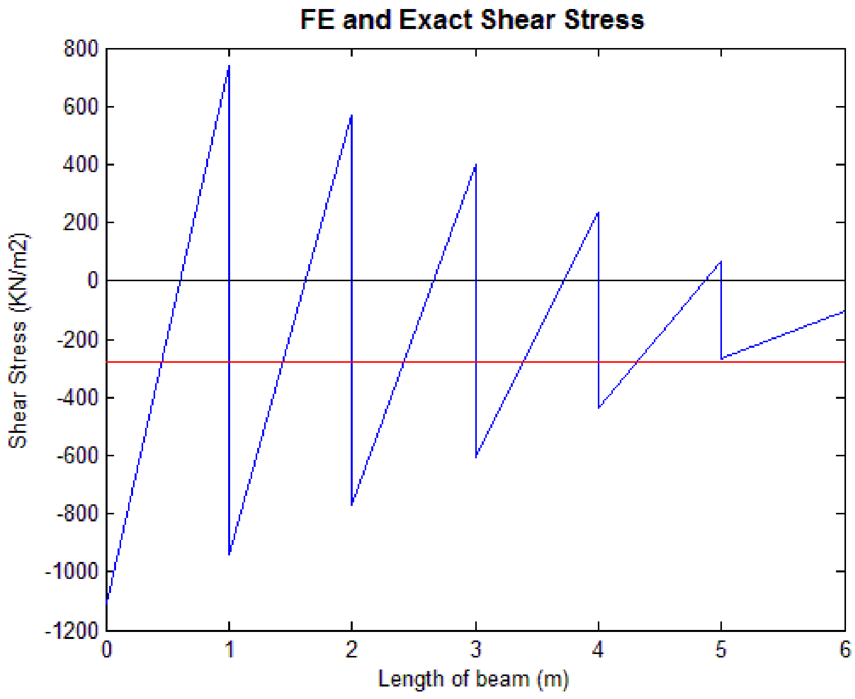
**Full Integration** It is defined as the quadrature rule of sufficient accuracy to integrate exactly the stiffness matrix of an undistorted element. According to Eq.3.11, in Q4 element, [**B**] is function of first order of $\xi$ and $\eta$, so [**K**] is a function of second orders of $\xi$ and $\eta$ and two gauss points are sufficient to get the exact integration (as mentioned in the previous paragraph). While for Q8 element, [**B**] is function of second orders of $\xi$ and $\eta$, so [**K**] is a fourth order function of $\xi$ and $\eta$. In this case, three gauss points are sufficient to get the exact integration.



(a) v = $-8.54*10^{-4}$ m

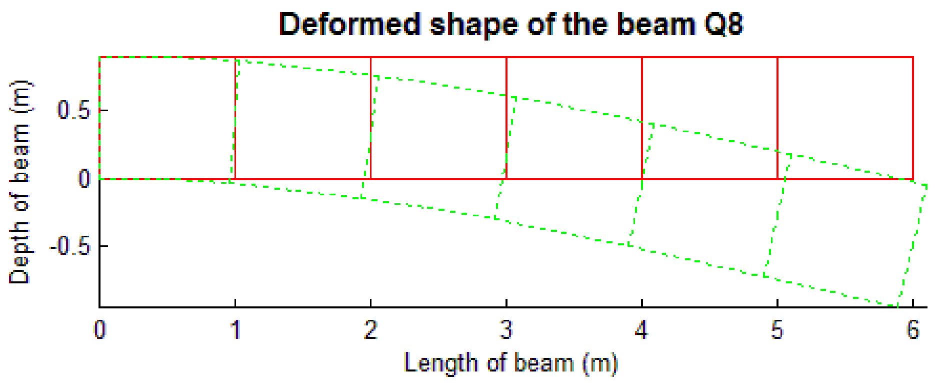


(b) $\sigma_x$ = 2110 KN/m$^2$

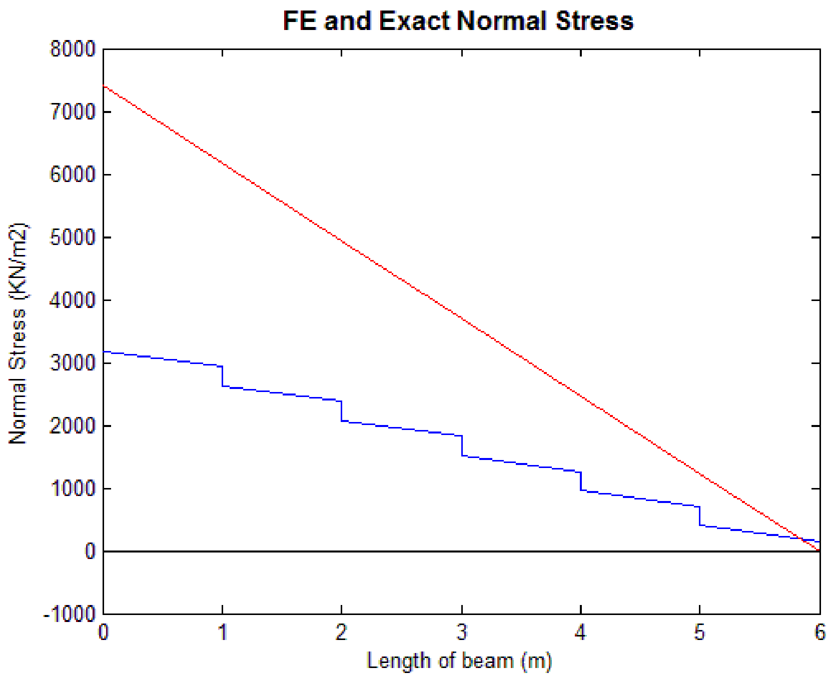


(c)

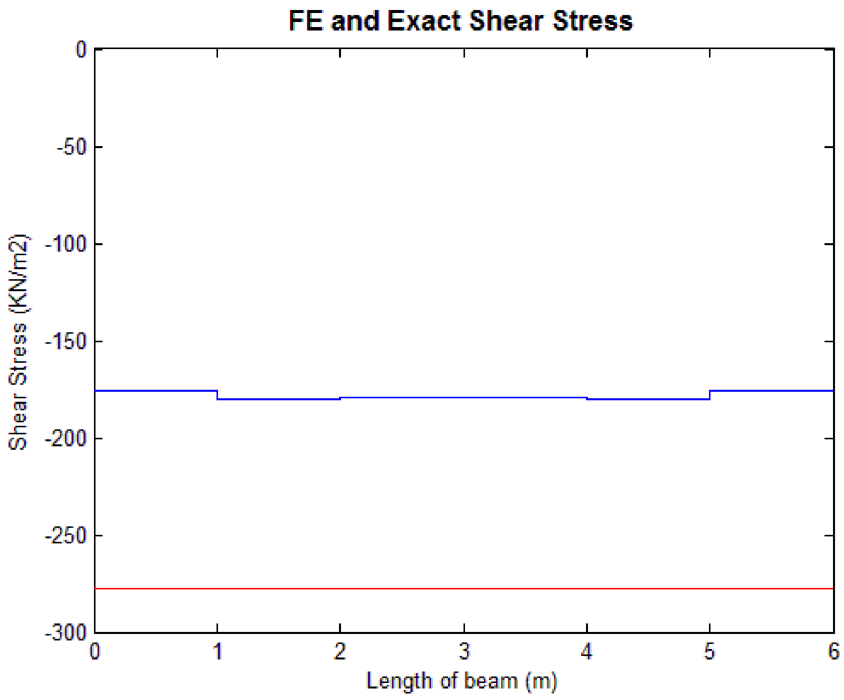
**Fig. 5-1** Analysis of Cantilever_C2_01_03_Q4 modeled by Full Integration



(a) v= -9.5104e-04 m



(b) $\sigma_x = 3183$ KN/m$^2$



(c)

**Fig. 5-2** Analysis of Cantilever_C2_01_03_Q8 modeled by Full Integration

Fig.5.1 shows the analysis of the cantilever done in the previous sections by Q4 element is now modeled by Full Integration of stiffness matrix. The results are quite the same of the exact integration done by MatLab. The same for Fig.5.2, which displays the cantilever beam previously done by Q8 element, is now analyzed by Full Integration of stiffness matrix. The results are exactly the same as if it is done by the exact integration method.

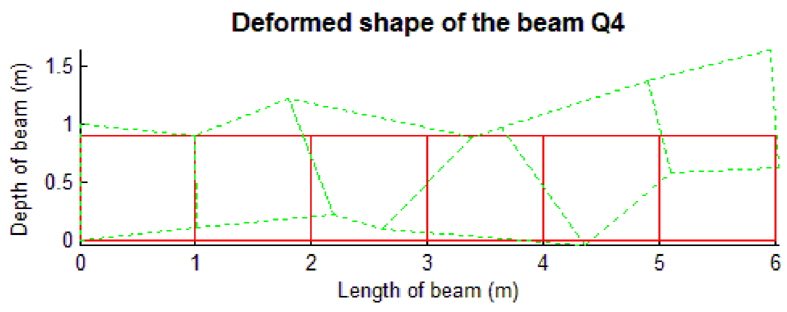
| | | **Exact** | **Q4 element** | **Full Integration** |
|---|---|---|---|---|
| **Normal stress (KN/m$^2$)** | 6 elements | 7407 | 2133 | 2110 |
| **Deflection (m)** | 6 elements | $-9.406*10^{-4}$ | $-6.073*10^{-4}$ | $-6.0783*10^{-4}$ |

**Table 5-1** Results of the cantilever beam modeled by Full integration method

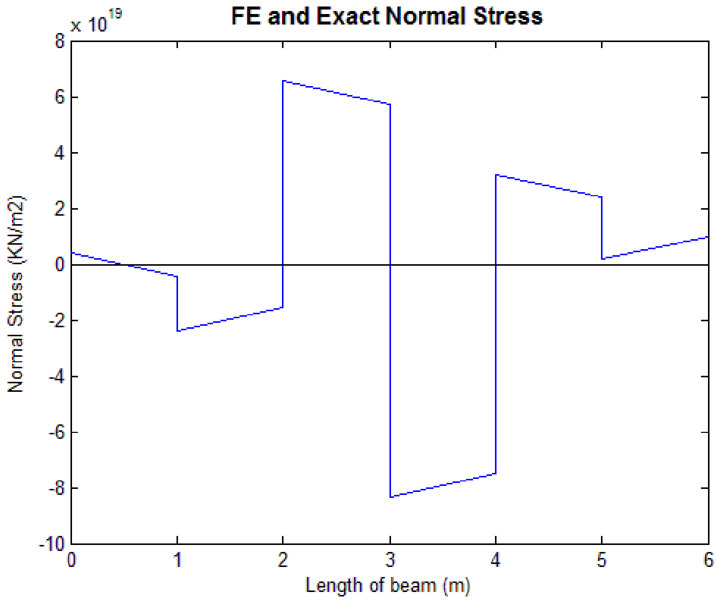
**Under-integration** is the use of gauss points less than the full order. It reduces the computation times especially in non-linear and dynamic analysis. It also improves the accuracy of the FE results by offsetting the over-stiffness associated with the 4-node element -as proven in section 4.1.2 and 4.1.3- as a result of its softening effect. Some polynomial terms vanish at gauss points and they don't make contribution in the strain energy. In mathematics, under-integration is called *rank deficient* [5, p. 223].

However, this method may cause various spurious deformation modes. The stiffness matrix of an element which incorporates a spurious mode has no resistance to the nodal loads. The number of the spurious mode can be determined as follow: In Fig.5.2, the cantilever beam is modeled using full integration (2 gauss points because it is a Q4 element), each element has eight d.o.f. The rank of the stiffness matrix for this element is five. The number of spurious mode can be calculated as the difference between the d.o.f. of the element (the order of the stiffness matrix) and its rank; which in this case is three; the rigid body motions (two translations and one rotation). For Fig.5.3, the cantilever beam is analyzed using one gauss point. The rank of $\mathbf{k^e}$ is three, so the number of spurious mode is 8-3=5 modes. There are two additional spurious modes may or may not lead to the singularity of $\mathbf{k^e}$. If $\mathbf{k^e}$ is singular, no possible solution is given. If $\mathbf{k^e}$ is nearly singular, the spurious modes will cause instability to the solution as shown in Fig.5.3a [14, p. 239&240].

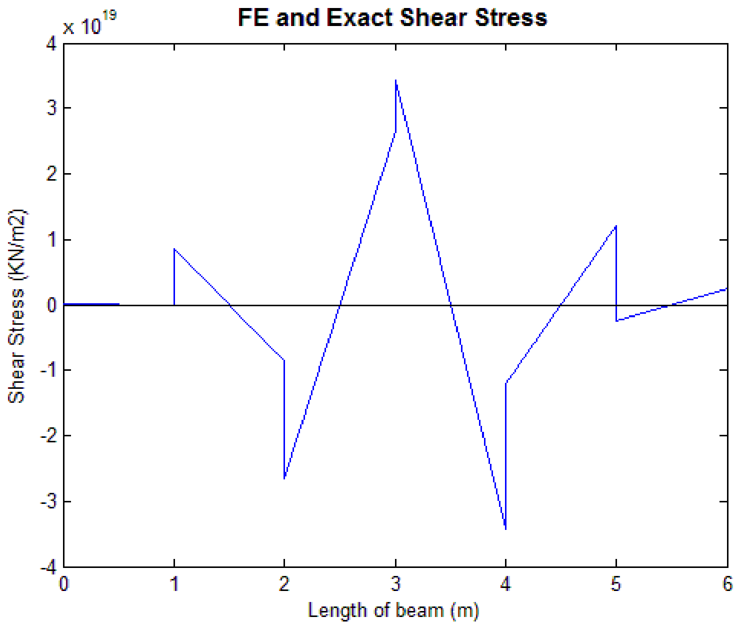
It is obvious in Fig.5.3a that the elements incorporate spurious mode and they cannot resist the nodal loads (this deformed shape is scaled; the real deformation is $2.78\times10^{8}$ m!). Also the normal and shear stresses are too large as shown in Fig.5.3b&c.



(a)



(b)



(c)

**Fig. 5-3** Analysis of Cantilever_C2_01_03_Q4 modeled by Under-Integration

# CHAPTER 6

CONCLUSION

# 6. Conclusion

Mainly the problems of FEM are about reaching the exact analysis in efficient and simple way. It has been proven in this paper that numerical analysis (mathematics generally) plays the main role to get the exact solution. The more you describe the elements in details (elements with more nodes), the more the results approach to the exact solution. But, it needs to solve complicated equations and analysis will take more time.

However, researches searched for a solution to the problem from another point of view. They studied the behavior of the elements. They found that it doesn’t behave according to the classical beam theory says. From then, the shear locking problem arises. This paper had shown different approaches to fix this problem.

Another problem related to the FEM is volumetric locking problem. Incompressible materials cannot be handled in this method. Many researches had been performed to find solutions for this problem as explained earlier in this paper.

The engineers –with the aid of mathematicians- are doing great efforts to search for other ways to study the behavior of the elements. Examples for these new methods are the boundary element method and the meshless method. Each of them has its advantages and disadvantages.

# CHAPTER 7

APPENDICES

# 7. Appendices

## 7.1 Appendix A: Exact Beam Analysis

### *7.1.1 CantileverBeam_C1*

```
%.......................................................................
%CantileverBeam_C1
%This file calculates the Bending Moment, shear force Diagram of a
%Cantilever beam. It also calculates the normal and shear stress. The
%cantilever beam is subjected to a concentrated load at its free end.
%This analysis is done using Beam Element.
%All units are in KN and m.

%Properties of the Material and Cross Section
E = 210e6;  %Modulus of Elasticity E
Lx = 6; %Length of the beam
Ly = 0.9;   %Depth
t = 0.3;    %Thickness
I=(t*Ly^3)/12;  %Moment of Inertia
A = .9*.3;  %Area of the cross section

%This cantilever beam is divided into six beam elements, each of
%length odx.
odx = 1;

%Calculation of stiffness matrix ok
ok1 = BeamElementStiffness(E,I,odx);
ok2 = BeamElementStiffness(E,I,odx);
ok3 = BeamElementStiffness(E,I,odx);
ok4 = BeamElementStiffness(E,I,odx);
ok5 = BeamElementStiffness(E,I,odx);
ok6 = BeamElementStiffness(E,I,odx);

%Assembling the Global Stiffness Matrix
%There are 7 nodes in the beam and each node has 2 d.o.f., so the
global stiffness matrix oK is 14*14
oK = zeros(14,14);
oK = BeamAssemble(oK,ok1,1,2);
oK = BeamAssemble(oK,ok2,2,3);
oK = BeamAssemble(oK,ok3,3,4);
oK = BeamAssemble(oK,ok4,4,5);
oK = BeamAssemble(oK,ok5,5,6);
oK = BeamAssemble(oK,ok6,6,7);

%Boundary Conditions
%This Cantilever beam is fixed-free, so neither vertical nor
%horizontal motions exist in the left end of the beam, which means
%u1=0 and u2=0. For the force vector, f1 and f2 are unknowns.
%All the elements in the force vector are zeros except f14 is -50.
of = zeros(14,1);   %Force vector
of(13,1) = -50;
of(2,:) = [];
of(1,:) = [];
ok = oK;
ok(2,:) = [];
ok(1,:) = [];
ok(:,2) = [];
ok(:,1) = [];
ou = ok\of;
oU = InsertZeroElement([1;2],ou); %The displacement vector oU
oF = oK*oU; %The Final force vector oF

%Displacement Vector for each element
ou1 = [oU(1) ; oU(2) ; oU(3) ; oU(4)];
ou2 = [oU(3) ; oU(4) ; oU(5) ; oU(6)];
ou3 = [oU(5) ; oU(6) ; oU(7) ; oU(8)];
ou4 = [oU(7) ; oU(8) ; oU(9) ; oU(10)];
ou5 = [oU(9) ; oU(10) ; oU(11) ; oU(12)];
ou6 = [oU(11) ; oU(12) ; oU(13) ; oU(14)];

%Force vector for each element
of1 = BeamElementForces(ok1,ou1);
of2 = BeamElementForces(ok2,ou2);
of3 = BeamElementForces(ok3,ou3);
of4 = BeamElementForces(ok4,ou4);
of5 = BeamElementForces(ok5,ou5);
of6 = BeamElementForces(ok6,ou6);

%Drawing the Bending Moment Diagram
figure(1)
hold on
BeamElementMomentDiagram(of1,0,1);
BeamElementMomentDiagram(of2,1,2);
BeamElementMomentDiagram(of3,2,3);
BeamElementMomentDiagram(of4,3,4);
BeamElementMomentDiagram(of5,4,5);
BeamElementMomentDiagram(of6,5,6);
hold off

%Distribution of the Normal Stress
figure(2)
hold on
ExactNormalStress(t,Ly,of1,0,1);
ExactNormalStress(t,Ly,of2,1,2);
ExactNormalStress(t,Ly,of3,2,3);
ExactNormalStress(t,Ly,of4,3,4);
ExactNormalStress(t,Ly,of5,4,5);
ExactNormalStress(t,Ly,of6,5,6);
hold off

%Shear Force Diagram
figure(3)
hold on
BeamElementShearDiagram(of1,0,1);
BeamElementShearDiagram(of2,1,2);
BeamElementShearDiagram(of3,2,3);
BeamElementShearDiagram(of4,3,4);
BeamElementShearDiagram(of5,4,5);
BeamElementShearDiagram(of6,5,6);
hold off

%Distribution of Shear stresses along the centroid of each element
figure(4)
hold on
ExactShearStress(A,-of1,0,1);
ExactShearStress(A,-of2,1,2);
ExactShearStress(A,-of3,2,3);
ExactShearStress(A,-of4,3,4);
ExactShearStress(A,-of5,4,5);
ExactShearStress(A,-of6,5,6);
hold off
%.....................................................................
```

### 7.1.2 *SimpleBeam_A1*

```
%......................................................................
%SimpleBeam_A1
This file calculates the Bending Moment, shear force Diagram of a
%simple beam. It also calculates the normal and shear stress. The
%cantilever beam is subjected to a concentrated load at its mid span.
%This analysis is done using Beam Element.
%All units are in KN and m.

%Properties of the beam's material and section
E = 210e6;  %Modulus of Elasticity E in KN/m2
Lx = 6; %Length of beam
Ly = 0.9;   %Depth
t = 0.3;    %Thickness
I=(t*Ly^3)/12;  %Second moment of Inertia
A = 0.3*0.9;    %Area of the cross section

%This simple beam is divided into six beam elements, each of length
odx
odx = 1;

%Calculation of stiffness
ok1 = BeamElementStiffness(E,I,odx);
ok2 = BeamElementStiffness(E,I,odx);
ok3 = BeamElementStiffness(E,I,odx);
ok4 = BeamElementStiffness(E,I,odx);
ok5 = BeamElementStiffness(E,I,odx);
ok6 = BeamElementStiffness(E,I,odx);

%Assembling the Global Stiffness Matrix
%There are 7 nodes in the beam and each node has 2 d.o.f., %so the
global stiffness matrix K is 14*14
oK = zeros(14,14);
oK = BeamAssemble(oK,ok1,1,2);
oK = BeamAssemble(oK,ok2,2,3);
oK = BeamAssemble(oK,ok3,3,4);
oK = BeamAssemble(oK,ok4,4,5);
oK = BeamAssemble(oK,ok5,5,6);
oK = BeamAssemble(oK,ok6,6,7);

%Boundary Condition
%This beam is hinged-roller, so there is no vertical motion in y
%direction for both end sides, which means u1=0 and u13=0. For the
%force vector, only f1 and f13 are the unknowns. All the elements in
%the force vector are zero except f7=-50 KN.
of = zeros(14,1);   %force vector
of(7,1) = -50;
of(13,:) = [];
of(1,:) = [];
ok = oK;
ok(13,:) = [];
ok(1,:) = [];
ok(:,13) = [];
ok(:,1) = [];
ou = ok\of;
oU = [0;ou(1:11);0;ou(12)]; %The displacement vector oU
```

```
oF = oK*oU; %The Final force vector oF

%Displacement vector for each element
ou1 = [oU(1) ; oU(2) ; oU(3) ; oU(4)];
ou2 = [oU(3) ; oU(4) ; oU(5) ; oU(6)];
ou3 = [oU(5) ; oU(6) ; oU(7) ; oU(8)];
ou4 = [oU(7) ; oU(8) ; oU(9) ; oU(10)];
ou5 = [oU(9) ; oU(10) ; oU(11) ; oU(12)];
ou6 = [oU(11) ; oU(12) ; oU(13) ; oU(14)];

%Force vector for each element
of1 = BeamElementForces(ok1,ou1);
of2 = BeamElementForces(ok2,ou2);
of3 = BeamElementForces(ok3,ou3);
of4 = BeamElementForces(ok4,ou4);
of5 = BeamElementForces(ok5,ou5);
of6 = BeamElementForces(ok6,ou6);

%Drawing the Bending Moment Diagram
figure(1)
hold on
BeamElementMomentDiagram(of1,0,1);
BeamElementMomentDiagram(of2,1,2);
BeamElementMomentDiagram(of3,2,3);
BeamElementMomentDiagram(of4,3,4);
BeamElementMomentDiagram(of5,4,5);
BeamElementMomentDiagram(of6,5,6);
hold off

%Distribution of the Normal Stress
figure(2)
hold on
ExactNormalStress(t,Ly,of1,0,1);
ExactNormalStress(t,Ly,of2,1,2);
ExactNormalStress(t,Ly,of3,2,3);
ExactNormalStress(t,Ly,of4,3,4);
ExactNormalStress(t,Ly,of5,4,5);
ExactNormalStress(t,Ly,of6,5,6);
hold off

%Shear Force Diagram
figure(3)
hold on
BeamElementShearDiagram(of1,0,1);
BeamElementShearDiagram(of2,1,2);
BeamElementShearDiagram(of3,2,3);
BeamElementShearDiagram(of4,3,4);
BeamElementShearDiagram(of5,4,5);
BeamElementShearDiagram(of6,5,6);
hold off

% Distribution of Shear stresses along the centroid of each element
figure(4)
hold on
ExactShearStress(A,-of1,0,1);
ExactShearStress(A,-of2,1,2);
ExactShearStress(A,-of3,2,3);
ExactShearStress(A,-of4,3,4);
ExactShearStress(A,-of5,4,5);
ExactShearStress(A,-of6,5,6);
hold off
%......................................................................
```

### 7.1.3 *Codes used in the script*

#### 7.1.3.1 *Beam Element Stiffness*

```
function k = BeamElementStiffness(E,I,L)
%BeamElementStiffness This function returns the element
% stiffness matrix for a beam element with modulus of elasticity E,
% moment of inertia I, and length L. The size of the element stiffness
% matrix is 4 x 4.
k = E*I/(L*L*L) * [12 6*L -12 6*L ; 6*L 4*L*L -6*L 2*L*L ;
-12 -6*L 12 -6*L ; 6*L 2*L*L -6*L 4*L*L];
end
```

#### 7.1.3.2 *Beam Assemble*

```
function K = BeamAssemble(K,k,i,j)
%BeamAssemble This function assembles the element stiffness
% matrix k of the beam element with nodes i and j into the global
%stiffness matrix K. This function returns the global stiffness
% matrix K after the element stiffness matrix k is assembled.
K(2*i-1,2*i-1) = K(2*i-1,2*i-1) + k(1,1);
K(2*i-1,2*i) = K(2*i-1,2*i) + k(1,2);
K(2*i-1,2*j-1) = K(2*i-1,2*j-1) + k(1,3);
K(2*i-1,2*j) = K(2*i-1,2*j) + k(1,4);
K(2*i,2*i-1) = K(2*i,2*i-1) + k(2,1);
K(2*i,2*i) = K(2*i,2*i) + k(2,2);
K(2*i,2*j-1) = K(2*i,2*j-1) + k(2,3);
K(2*i,2*j) = K(2*i,2*j) + k(2,4);
K(2*j-1,2*i-1) = K(2*j-1,2*i-1) + k(3,1);
K(2*j-1,2*i) = K(2*j-1,2*i) + k(3,2);
K(2*j-1,2*j-1) = K(2*j-1,2*j-1) + k(3,3);
K(2*j-1,2*j) = K(2*j-1,2*j) + k(3,4);
K(2*j,2*i-1) = K(2*j,2*i-1) + k(4,1);
K(2*j,2*i) = K(2*j,2*i) + k(4,2);
K(2*j,2*j-1) = K(2*j,2*j-1) + k(4,3);
K(2*j,2*j) = K(2*j,2*j) + k(4,4);
end
```

#### 7.1.3.3 *Beam Assemble*

```
function [ newvec ] = InsertZeroElement( ind,oldvec )
%InsertZeroElement This function inserts zero elements in an ascending
%order in a vector. It returns a new vector after inserting zero
%elements. The input is ind which is the indices of the zeros and
%oldvec which is the vector needed to put zeros inside.

newvec = oldvec;
rowsa = sort(ind,'ascend'); %arrangement of the indices in ascending
order

for j = 1: size(rowsa,1)
    for m = 1: size(oldvec)
        if m >= rowsa(j)
            newvec(rowsa(j)) = 0;
            newvec(m+1) = oldvec(m);
        else
            newvec(m) = oldvec(m);
        end
    end
    oldvec = newvec;
end
end
```

#### *7.1.3.4 Beam Forces*

```
function y = BeamElementForces(k,u)
%BeamElementForces This function returns the element nodal force
% vector given the element stiffness matrix k and the element nodal
%displacement vector u.
y = k * u;
end
```

#### *7.1.3.5 Beam Element Moment Diagram*

```
function y = BeamElementMomentDiagram(f, L1, L2)
%BeamElementMomentDiagram This function plots the bending moment
% diagram for the beam element with nodal force vector f and length L1
%and L2.
x = [L1 ; L2];
z = [-f(2) ; f(4)];
hold on;
title('Bending Moment Diagram');
plot(x,z);
y1 = [0 ; 0];
plot(x,y1,'k')
end
```

#### *7.1.3.6 Exact Normal Stress*

```
function y = ExactNormalStress( t,Ly,f,L1,L2 )
%ExactNormalStress This function returns a graph for the normal
%stresses along the length of the beam. The inputs are the thickness
of %the beam t, its depth Ly, force vector f and the increment L1&L2.
I=t*Ly^3/12;
x=[L1;L2];
z=[(f(2)*Ly/2)/I;(-f(4)*Ly/2)/I];
hold on;
title('FE and Exact Normal Stress','FontSize',12,'FontWeight','bold');
plot(x,z,'r');
y1=[0;0];
plot(x,y1,'k');
xlabel('Length of beam (m)')
ylabel('Normal Stress (KN/m2)')
end
```

#### *7.1.3.7 Exact Shear Stress*

```
function y = BeamElementShearDiagram(f, L1, L2)
```

```
%BeamElementShearDiagram This function plots the shear force diagram
%for the beam element with nodal force vector f and length L1 and L2.
x = [L1 ; L2];
z = [f(1) ; -f(3)];
hold on;
title('Shear Force Diagram');
plot(x,z);
y1 = [0 ; 0];
plot(x,y1,'k')
end
```

## 7.2 Appendix B: CST & LST

### *7.2.1 CST*

```
%..................................................................
%CantileverBeam_C2_01_01_CST

%This script calculates the stresses of a cantilever beam supported by
%concentrated load at its free end using Constant Strain Triangular
%element.
%All units are in metric KN m

%Clear memory
clear all
close all
clc

%Properties of the Material and Cross Section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimensions of the beam
Lx = 6; %Length of the beam
Ly = 0.9;   %Depth
t = 0.3;    %thickness

%Generation of nodes
AutoArrangeCoordinates

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
K = zeros(14*2,14*2);
no = 0;
for cc = 1: 12
    iii = ii(cc,1); %node i
    jjj = jj(cc,1); %node j
    mm = m(cc,1);   %node m
    x11 = x1(cc,1); %x coordinates of node i
    y11 = y1(cc,1); %y coordinates of node i
    x22 = x2(cc,1); %x coordinates of node j
    y22 = y2(cc,1); %y coordinates of node j
    x33 = x3(cc,1); %x coordinates of node m
    y33 = y3(cc,1); %y coordinates of node m
    %Ae is the area of the element. It must be positive number.
    Ae(cc,1) = LinearTriangleElementArea(x11,y11,x22,y22,x33,y33);
    K =
LinearTriangleElementStiffness(E,NU,t,x11,y11,x22,y22,x33,y33,1);
    no = no + 1;
    K = LinearTriangleAssemble(K,k,iii,jjj,mm);
end

%Boundary Conditions
f = zeros(size(K),1);   %f is the force vector
f(28,1) = -25;
f(14,1) = -25;
f(15,:) = [];
f(2,:) = [];
f(1,:) = [];
```

```
k = K;
k(15,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,15) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1 ;2 ;15],u);    %the displacement vector
F = K*U;    %The final force vector

%Calculation of the stresses sigma
no = 0;
for vv = 1: 12
    uu((6*no)+1:(6*no)+6,:) = [ U(((ii(vv,1))*2)-1);
    U(((ii(vv,1))*2)); U(((jj(vv,1))*2)-1); U(((jj(vv,1))*2));
    U(((m(vv,1))*2)-1); U(((m(vv,1))*2))];
    sigma((3*no)+1:(3*no)+3,:) =
    LinearTriangleElementStresses(E,NU,x1(vv,1),y1(vv,1),x2(vv,1),y2(v
    v,1),x3(vv,1),y3(vv,1),1,uu((6*no)+1:(6*no)+6,:));
    no = no + 1;
end

%Normal Stresses
s = sigma(1:3:size(sigma),:);
no = 0;
for i = 1: 11
    s1(no+1,1) = s(i,1);
    s1(no+2,1) = s(i+1,1);
    no = no + 2;
end
s1 = [s(1,1); s1; s(12,1)];
x = [0; .5;.5;1; 1;1.5;1.5; 2; 2;2.5;2.5; 3; 3;3.5;3.5; 4; 4;4.5; 4.5;
5; 5;5.5;5.5; 6];
figure(2)
plot(x,s1)

%Shear Stresses
q = sigma(3:3:size(sigma),:);   %the shear stress
no = 0;
for i = 1: 11
    q1(no+1,1) = q(i,1);
    q1(no+2,1) = q(i+1,1);
    no = no + 2;
end
q1 = [q(1,1); q1; q(12,1)];
figure(4)
plot(x,q1)

%Deformed shape of the beam
figure(5)
DeformedBeamCST(U*1000,element)

%the exact analysis
CantileverBeam_C1
%...................................................................
```

### *7.2.2 LST*

```
%.......................................................................
%CantileverBeam_C2_01_02_LST

%This script calculates the stresses of a cantilever beam supported by
%concentrated load at its free end using Linear Strain Triangular
%element
%All units are in metric KN m

%Clear memory
clear all
close all
clc

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson’s ratio

%Dimensions of the beam
Lx = 6; %Length of the beam
Ly = 0.9;   %Depth
t = 0.3;    %thickness

%Generation of nodes
AutoArrangeCoordinates

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
K = zeros(39*2,39*2);
no = 0;
for cc = 1: 12
    iii = ii(cc,1); %node i
    jjj = jj(cc,1); %node j
    mm = m(cc,1);   %node m
    pp = p(cc,1);   %node p
    qq = q(cc,1);   %node q
    rr = r(cc,1);   %node r
    x11 = x1(cc,1); %x coordinate of node i
    y11 = y1(cc,1); %y coordinate of node i
    x22 = x2(cc,1); %x coordinate of node j
    y22 = y2(cc,1); %y coordinate of node j
    x33 = x3(cc,1); %x coordinate of node m
    y33 = y3(cc,1); %y coordinate of node m
    %Ae is the area of an element and it must be positive
    Ae(cc,1) = QuadTriangleElementArea(x11,y11,x22,y22,x33,y33);
    k =QuadTriangleElementStiffness(E,NU,t,x11,y11,x22,y22,x33,y33,1);
    no = no + 1;
    K = QuadTriangleAssemble(K,k,iii,jjj,mm,pp,qq,rr);
end

%Boundary Conditions
f = zeros(size(K),1);   %f is the force vector
f(28,1) = -25;
f(14,1) = -25;
f(42,:) = [];
f(41,:) = [];
f(16,:) = [];
f(15,:) = [];
f(2,:) = [];
f(1,:) = [];
```

```
k = K;
k(42,:) = [];
k(41,:) = [];
k(16,:) = [];
k(15,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,42) = [];
k(:,41) = [];
k(:,16) = [];
k(:,15) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
%the displacement vector
U = InsertZeroElement([1 ;2 ;15 ;16 ;41 ;42],u);
F = K*U;    %The final force vector

%Calculation of the stresses sigma
no = 0;
for vv = 1: 12    %12 elements
    uu((12*no)+1:(12*no)+12,:) = [ U(((ii(vv,1))*2)-1);
    U(((ii(vv,1))*2)); U(((jj(vv,1))*2)-1); U(((jj(vv,1))*2));
    U(((m(vv,1))*2)-1); U(((m(vv,1))*2)); U(((p(vv,1))*2)-1);
    U(((p(vv,1))*2)); U(((q(vv,1))*2)-1); U(((q(vv,1))*2));
    U(((r(vv,1))*2)-1); U(((r(vv,1))*2))];
    sigma((3*no)+1:(3*no)+3,:) =
    QuadTriangleElementStresses(E,NU,x1(vv,1),y1(vv,1),x2(vv,1),y2(vv,
    1),x3(vv,1),y3(vv,1),1,uu((12*no)+1:(12*no)+12,:));
    no = no + 1;
end

%Normal Stresses
s = sigma(1:3:size(sigma),:);
no = 0;
for i = 1: 11
    s1(no+1,1) = s(i,1);
    s1(no+2,1) = s(i+1,1);
    no = no + 2;
end
s1 = [s(1,1); s1; s(12,1)];
x = [0; .5; .5; 1; 1; 1.5; 1.5; 2; 2; 2.5; 2.5; 3; 3; 3.5; 3.5; 4; 4;
4.5; 4.5; 5; 5; 5.5; 5.5; 6];
figure(2)
plot(x,s1)

%Shear Stresses
q = sigma(3:3:size(sigma),:);
no = 0;
for i = 1: 11
    q1(no+1,1) = q(i,1);
    q1(no+2,1) = q(i+1,1);
    no = no + 2;
end
q1 = [q(1,1); q1; q(12,1)];
figure(4)
plot(x,q1)
%Deformed shape of the beam
figure(5)
DeformedBeamLST(U*1000)
CantileverBeam_C1
%.....................................................................
```

### *7.2.3 Codes used in the script*

#### *7.2.3.1 Auto Arrange Coordinates*

```
%This script is to auto-arrange the coordinates of the cantilever beam
%for CST & LST problem only for this problem. The output is a matrix
%called element matrix which has all the %details required for the
%element; the id of the nodes and their positions
ni = 1; %node i
nj = 2; %node j
nm = 8; %node m
np = 15;    %node p
npp = 22;
nq = 22;    %node q
nr = 21;    %node r
nrr = 34;
nx1 = 0;    %x coordinate of node i
nx2 = 1;    %x coordinate of node j
nx3 = 0;    %x coordinate of node m
for c = 1:2:12
    ii(c) = ni;
    ii(c + 1) = 7 + ni;
    ni = ni + 1;

    jj(c) = nj;
    jj(c + 1) = nj;
    nj = nj + 1;

    m(c) = nm;
    m(c + 1) = nm + 1;
    nm = nm + 1;

    p(c) = np;
    p(c + 1) = npp;
    np = np + 1;
    npp = npp + 2;

    q(c) = nq;
    q(c + 1) = nq + 1;
    nq = nq + 2;

    r(c) = nr;
    r(c + 1) = nrr;
    nr = nr + 2;
    nrr = nrr + 1;

    x1(c) = nx1;
    x1(c + 1) = nx1;
    nx1 = nx1 + 1;

    y1(c) = 0;
    y1(c + 1) = 0.9;

    x2(c) = nx2;
    x2(c + 1) = nx2;
    nx2 = nx2 + 1;

    y2(c) = 0;
    y2(c + 1) = 0;
    x3(c) = nx3;
    x3(c + 1) = nx3 + 1;
    nx3 = nx3 + 1;

    y3(c) = 0.9;
    y3(c + 1) = 0.9;
end
ii = ii';
jj = jj';
m = m';
p = p';
q = q';
r = r';
x1 = x1';
y1 = y1';
x2 = x2';
y2 = y2';
x3 = x3';
y3 = y3';
element = [ii jj m p q r x1 y1 x2 y2 x3 y3];
```

#### 7.2.3.2 *Linear Triangle Element Area*

```
function y = LinearTriangleElementArea(xi,yi,xj,yj,xm,ym)
%LinearTriangleElementArea This function returns the area of the
% linear triangular element whose first node has coordinates (xi,yi),
% second node has coordinates (xj,yj), and third node has coordinates
% (xm,ym).
y = (xi*(yj-ym) + xj*(ym-yi) + xm*(yi-yj))/2;
end
```

#### 7.2.3.3 *Linear Triangle Element Stiffness*

```
function y =
LinearTriangleElementStiffness(E,NU,t,xi,yi,xj,yj,xm,ym,p)
%LinearTriangleElementStiffness This function returns the element
%stiffness matrix for a linear triangular element with modulus of
%elasticity E, Poisson’s ratio NU, thickness t, coordinates of the
%first node (xi,yi), coordinates of the second node (xj,yj), and
%coordinates of the third node(xm,ym). Use p = 1 for cases of plane
%stress, and p = 2 for cases of plane strain. The size of the element
%stiffness matrix is 6 x 6.
A = (xi*(yj-ym) + xj*(ym-yi) + xm*(yi-yj))/2;
betai = yj-ym;
betaj = ym-yi;
betam = yi-yj;
gammai = xm-xj;
gammaj = xi-xm;
gammam = xj-xi;
B = [betai 0 betaj 0 betam 0;
    0 gammai 0 gammaj 0 gammam;
    gammai betai gammaj betaj gammam betam]/(2*A);
if p == 1
    D = (E/(1-NU*NU))*[1 NU 0; NU 1 0; 0 0 (1-NU)/2];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU NU 0; NU 1-NU 0; 0 0 (1-2*NU)/2];
end
y = t*A*B'*D*B;
```

```
end
```

#### 7.2.3.4 *Linear Triangle Assemble*

```
function y = LinearTriangleAssemble( K,k,i,j,m )
%LinearTriangleAssemble This function assembles the element
% stiffness matrix k of the linear triangular element with nodes i, j,
% and m into the global stiffness matrix K.
% This function returns the global stiffness matrix K after the
%element stiffness matrix k is assembled.
K(2*i-1,2*i-1) = K(2*i-1,2*i-1)+k(1,1);
K(2*i-1,2*i) = K(2*i-1,2*i)+k(1,2);
K(2*i-1,2*j-1) = K(2*i-1,2*j-1)+k(1,3);
K(2*i-1,2*j) = K(2*i-1,2*j)+k(1,4);
K(2*i-1,2*m-1) = K(2*i-1,2*m-1)+k(1,5);
K(2*i-1,2*m) = K(2*i-1,2*m)+k(1,6);
K(2*i,2*i-1) = K(2*i,2*i-1)+k(2,1);
K(2*i,2*i) = K(2*i,2*i)+k(2,2);
K(2*i,2*j-1) = K(2*i,2*j-1)+k(2,3);
K(2*i,2*j) = K(2*i,2*j)+k(2,4);
K(2*i,2*m-1) = K(2*i,2*m-1)+k(2,5);
K(2*i,2*m) = K(2*i,2*m)+k(2,6);
K(2*j-1,2*i-1) = K(2*j-1,2*i-1)+k(3,1);
K(2*j-1,2*i) = K(2*j-1,2*i)+k(3,2);
K(2*j-1,2*j-1) = K(2*j-1,2*j-1)+k(3,3);
K(2*j-1,2*j) = K(2*j-1,2*j)+k(3,4);
K(2*j-1,2*m-1) = K(2*j-1,2*m-1)+k(3,5);
K(2*j-1,2*m) = K(2*j-1,2*m)+k(3,6);
K(2*j,2*i-1) = K(2*j,2*i-1)+k(4,1);
K(2*j,2*i) = K(2*j,2*i)+k(4,2);
K(2*j,2*j-1) = K(2*j,2*j-1)+k(4,3);
K(2*j,2*j) = K(2*j,2*j)+k(4,4);
K(2*j,2*m-1) = K(2*j,2*m-1)+k(4,5);
K(2*j,2*m) = K(2*j,2*m)+k(4,6);
K(2*m-1,2*i-1) = K(2*m-1,2*i-1)+k(5,1);
K(2*m-1,2*i) = K(2*m-1,2*i)+k(5,2);
K(2*m-1,2*j-1) = K(2*m-1,2*j-1)+k(5,3);
K(2*m-1,2*j) = K(2*m-1,2*j)+k(5,4);
K(2*m-1,2*m-1) = K(2*m-1,2*m-1)+k(5,5);
K(2*m-1,2*m) = K(2*m-1,2*m)+k(5,6);
K(2*m,2*i-1) = K(2*m,2*i-1)+k(6,1);
K(2*m,2*i) = K(2*m,2*i)+k(6,2);
K(2*m,2*j-1) = K(2*m,2*j-1)+k(6,3);
K(2*m,2*j) = K(2*m,2*j)+k(6,4);
K(2*m,2*m-1) = K(2*m,2*m-1)+k(6,5);
K(2*m,2*m) = K(2*m,2*m)+k(6,6);
y = K;
end
```

#### 7.2.3.5 *Linear Triangle Assemble*

```
function y = LinearTriangleElementStresses(E,NU,xi,yi,xj,yj,xm,ym,p,u)
%LinearTriangleElementStressesThis function returns the element
% stress vector for a linear triangular element with modulus of
% elasticity E, Poisson’s ratio NU, coordinates of the first node
%(xi,yi), coordinates of the second node (xj,yj), coordinates of the
%third node (xm,ym), and element displacement vector u. Use p = 1 for
%cases of plane stress, and p = 2 for cases of plane strain. The size
%of the element stress vector is 3 x 1.
A = (xi*(yj-ym)+xj*(ym-yi)+xm*(yi-yj))/2;
```

```
betai = yj-ym;
betaj = ym-yi;
betam = yi-yj;
gammai = xm-xj;
gammaj = xi-xm;
gammam = xj-xi;
B = [betai 0 betaj 0 betam 0;
    0 gammai 0 gammaj 0 gammam;
    gammai betai gammaj betaj gammam betam]/(2*A);
if p == 1
    D = (E/(1-NU*NU))*[1 NU 0; NU 1 0; 0 0 (1-NU)/2];
elseif p ==2
    D = (E/(1+NU)/(1-2*NU))*[1-NU NU 0; NU 1-NU 0; 0 0 (1-2*NU)/2];
end
y = D*B*u;
end
```

### 7.2.3.6 *Deformed Beam CST*

```
function y = DeformedBeamCST( U,element )
%DeformedBeamCST This function returns a deformed shape of a
%cantilever beam modeled by CST. The inputs are the displacement
%vector U and the element matrix which has all the details for the
%element.
hold on
title('Deformed Cantilever Beam modeled by
CST','FontSize',12,'FontWeight','bold')
xlabel('Length (m)')
ylabel('Displacement (m)*10^-3')
axis equal
for g = 1: size(element,1)
    i = element(g,1);   %node i
    j = element(g,2);   %node j
    m = element(g,3);   %node m
    x1 = element(g,7);  %x coordinate of node i
    y1 = element(g,8);  %y coordinate of node i
    x2 = element(g,9);  %x coordinate of node j
    y2 = element(g,10); %y coordinate of node j
    x3 = element(g,11); %x coordinate of node m
    y3 = element(g,12); %y coordinate of node m
    ux1 = U((i*2)-1,1); %displacement in x direction at node i
    ux2 = U((j*2)-1,1); %displacement in x direction at node j
    ux3 = U((m*2)-1,1); %displacement in x direction at node m
    uy1 = U((i*2),1);   %displacement in y direction at node i
    uy2 = U((j*2),1);   %displacement in y direction at node j
    uy3 = U((m*2),1);   %displacement in y direction at node m
    x1new = x1+ux1; %the new x coordinates of node i
    x2new = x2+ux2; %the new x coordinates of node j
    x3new = x3+ux3; %the new x coordinates of node m
    y1new = y1+uy1; %the new y coordinates of node i
    y2new = y2+uy2; %the new y coordinates of node j
    y3new = y3+uy3; %the new y coordinates of node m
    plot([x1, x2], [y1, y2]);
    plot([x2, x3], [y2, y3]);
    plot([x3, x1], [y3, y1]);
    plot([x1new, x2new], [y1new, y2new],':','color',[1 0 0]);
    plot([x2new, x3new], [y2new, y3new],':','color',[1 0 0]);
    plot([x3new, x1new], [y3new, y1new],':','color',[1 0 0]);
end
axis tight
hold off
end
```

## 7.3 Appendix C: Q4 element

### 7.3.1 *Cantilever_C2_01_03_Q4*

```
%.....................................................................
%Cantilever_C1_01_03_Q4
%This script calculates the normal and shear stresses of a cantilever
%beam loaded by 1 concentrated load at its edge. This beam is modeled
%by Q4 element.
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the Material and Cross Section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
f(28,1) = -25;
f(14,1) = -25;
f(15,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;  %k is the stiffness matrix
k(15,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,15) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([15;2;1],u);  %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (sx, sy and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses sx at the boundary nodes of the beam
%xB & yB are the positions of the sigmaBX stresses
%Qc is the shear stresses at the centroid of the beam
%Q0 is the min shear stresses which is at the top elements of the beam
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculation(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);
```

```
%Draw the normal stress
figure(2)
DrawStresses(Lx, dx, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeam(element)
DrawDeformedBeam(element,U*1000);
hold off

%Exact analysis
CantileverBeam_C1
%..................................................................
```

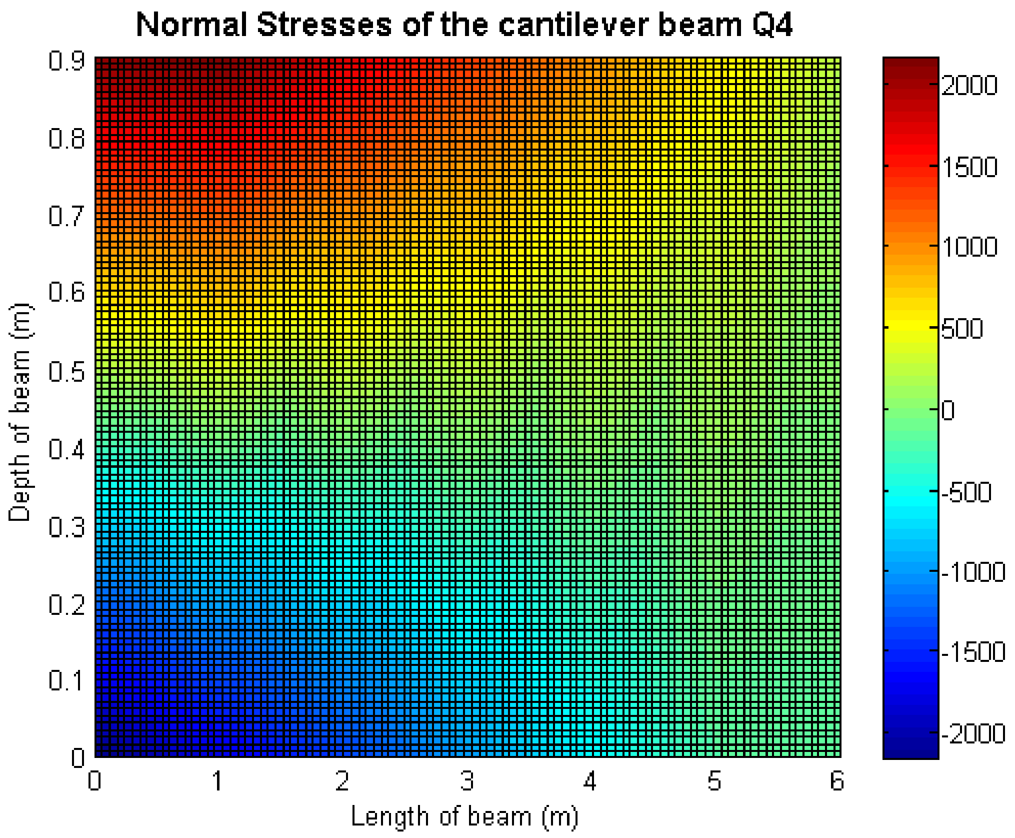


**Fig. 7-1** Normal Stresses in CantileverBeam_C2_01_03_Q4

### *7.3.2 Cantilever_C2_02_03_Q4*

The previous script is repeated. Only the meshing and Global Stiffness Matrix are changed to the following.

```
%Meshing
dx = 0.5; %Length of the element in x direction
dy = 0.225;   %Length of the element in y direction
```

```
%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessFinalGUIDE(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
%Dividing the nodal forces along the elements on the left side
f(65*2,1) = -50/(ny+1);
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 53;
%Removing all the unknown forces and the corresponding rows and
%columns in the global stiffness matrix
for i = 1: ny+1
f(no*2,:) = [];
f((no*2)-1,:) = [];
k(no*2,:) = [];
k((no*2)-1,:) = [];
k(:,no*2) = [];
k(:,(no*2)-1)=[];
noo((i*2),1) = no*2;
noo((i*2)-1,1) = (no*2)-1;
no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement(noo,u);
F = K*U;
```

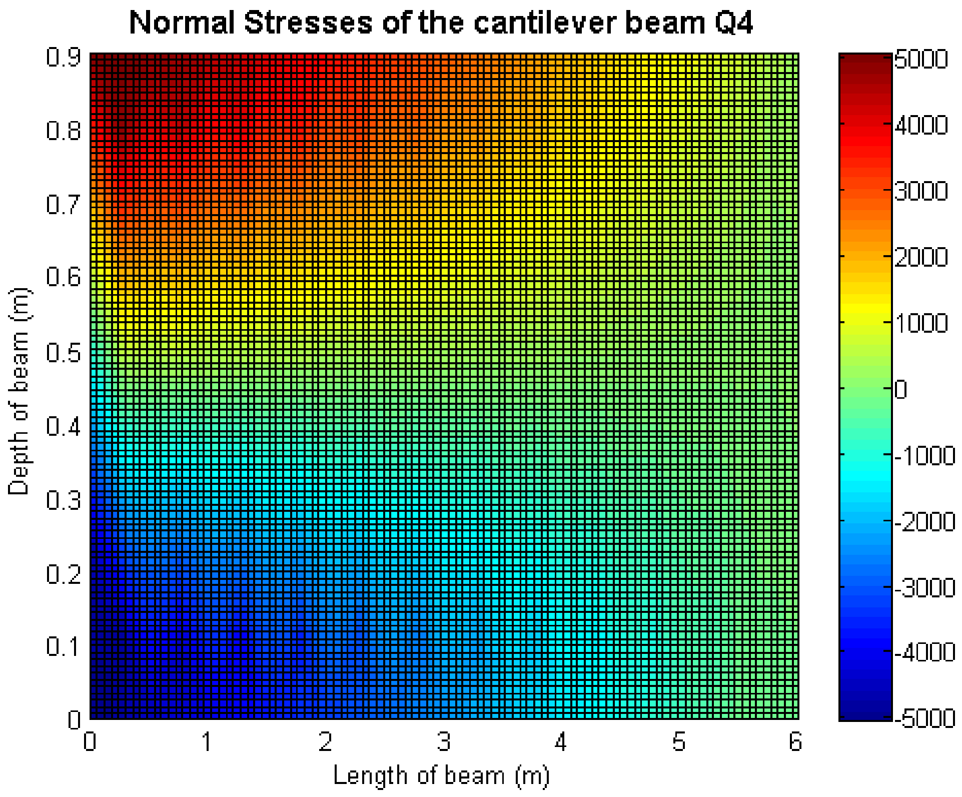


**Fig. 7-2** Normal Stress in CantileverBeam_C2_02_03_Q4

### *7.3.3 Cantilever_C2_03_03_Q4*

The previous script is repeated. Only the meshing and Global Stiffness Matrix are changed to the following.

```
%Meshing
dx = 0.1;   %Length of the element in x direction
dy = 0.05;  %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessFinalGUIDE(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
f(1159*2,1) = -50/(ny+1);
%Dividing the nodal forces along the elements on the left side
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 1099;
%Removing all the unknown forces and the corresponding rows and
%columns in the global stiffness matrix
for i = 1: ny+1
f(no*2,:) = [];
f((no*2)-1,:) = [];
k(no*2,:) = [];
k((no*2)-1,:) = [];
k(:,no*2) = [];
k(:,(no*2)-1)=[];
noo((i*2),1) = no*2;
noo((i*2)-1,1) = (no*2)-1;
no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement(noo,u);
F = K*U;
```

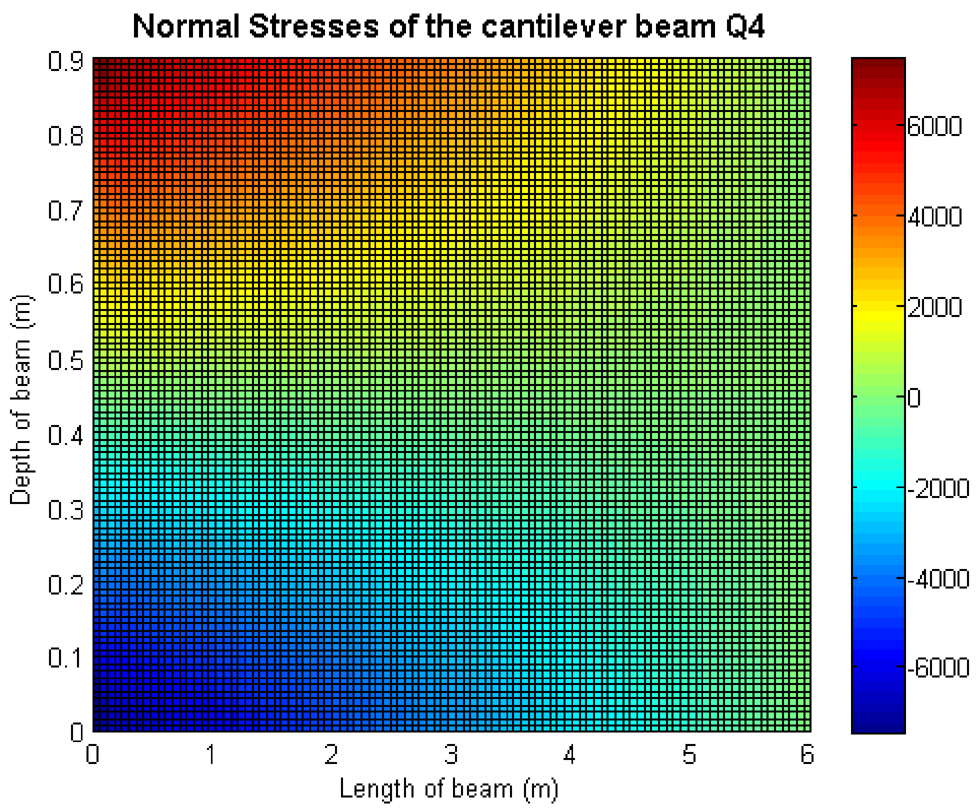
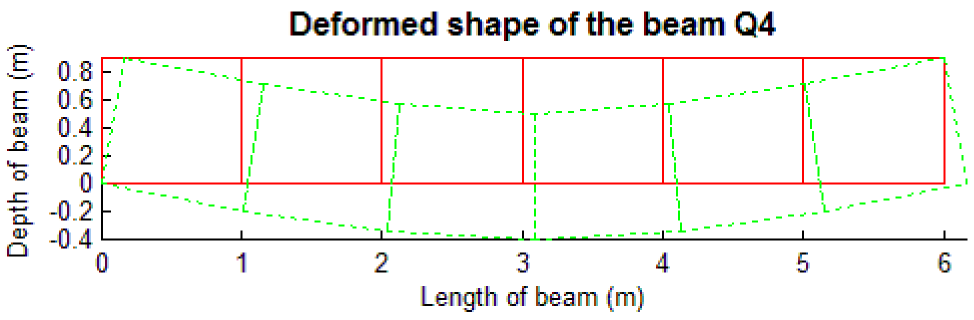


**Fig. 7-3** CantileverBeam_C2_03_03_Q4

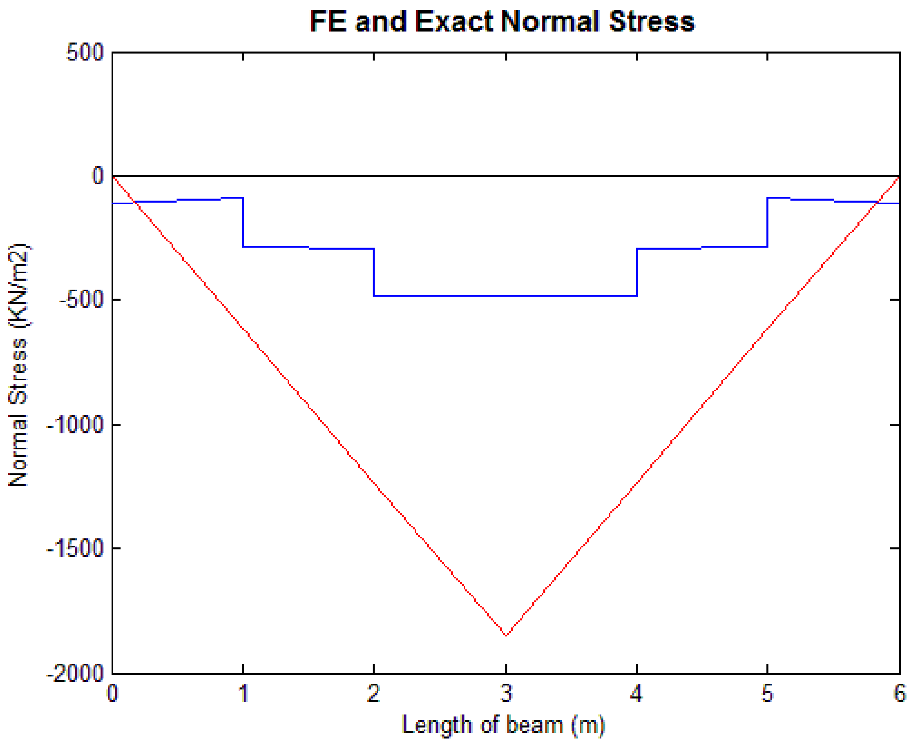
### *7.3.4 SimpleBeam_A1_01_03*

```
%...................................................................
%SimpleBeam_A2_01_03_Q4
%This script calculates the normal and shear stresses of a simple beam
%loaded by 1 concentrated load at mid-span
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
```

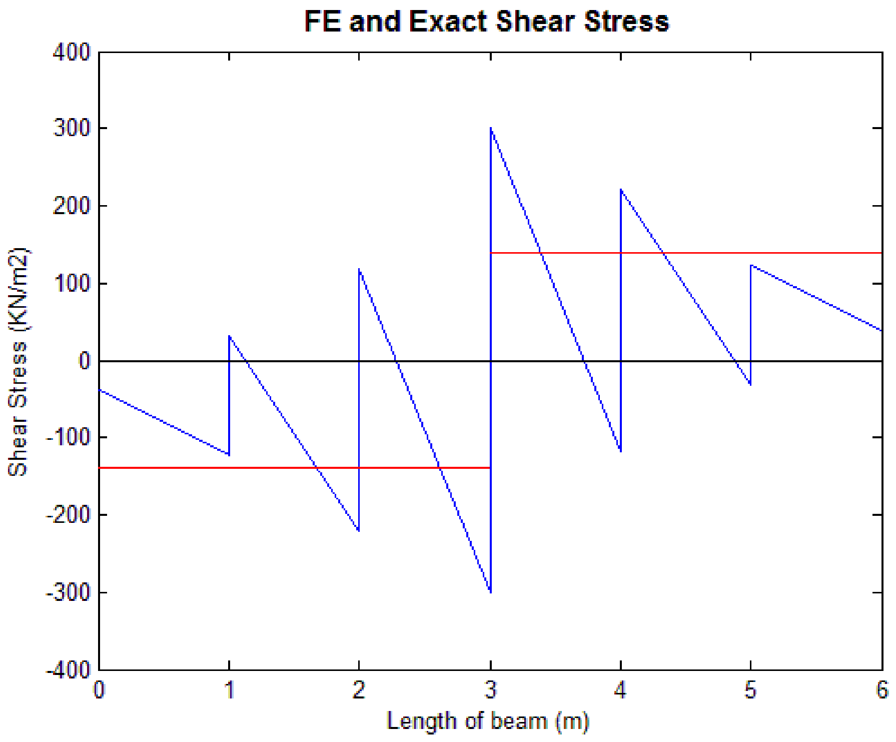
```
f(22,1) = -25;
f(8,1) = -25;
f(14,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(14,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,14) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([14;2;1],u);  %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (?x, ?y and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses ?x at the boundary nodes of the beam
%xB & yB are the positions of the sigmaBX stresses
%Qc is the shear stresses at the centroid of the beam
%Q0 is the min shear stresses which is at the top elements of the beam
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculation(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);

%Draw the normal stress
figure(2)
DrawStresses(Lx, dx,element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeam(element)
DrawDeformedBeam(element,U*1000);
hold off

%Exact analysis
SimpleBeam_A1
%.....................................................................
```


Deformed shape of the beam Q4
Depth of beam (m)
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
1
2
3
4
5
6
Length of beam (m)

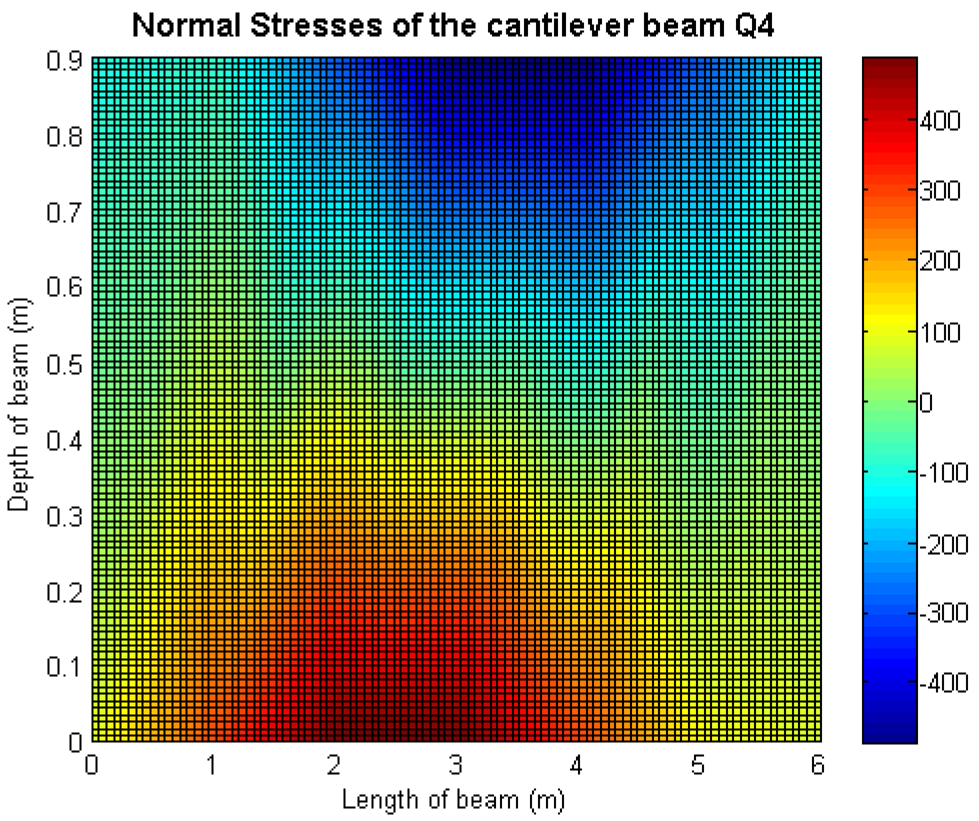
(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6
Length of beam (m)

(b)


FE and Exact Shear Stress
Shear Stress (KN/m2)
400
300
200
100
0
-100
-200
-300
-400
0
1
2
3
4
5
6
Length of beam (m)

(c)



(d)

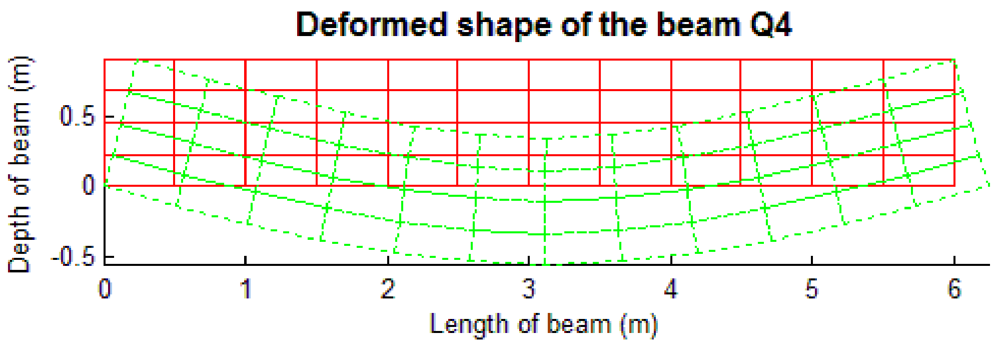
**Fig. 7-4** SimpleBeam_A1_01_03_Q4

### *7.3.5 SimpleBeam_A1_02_03*

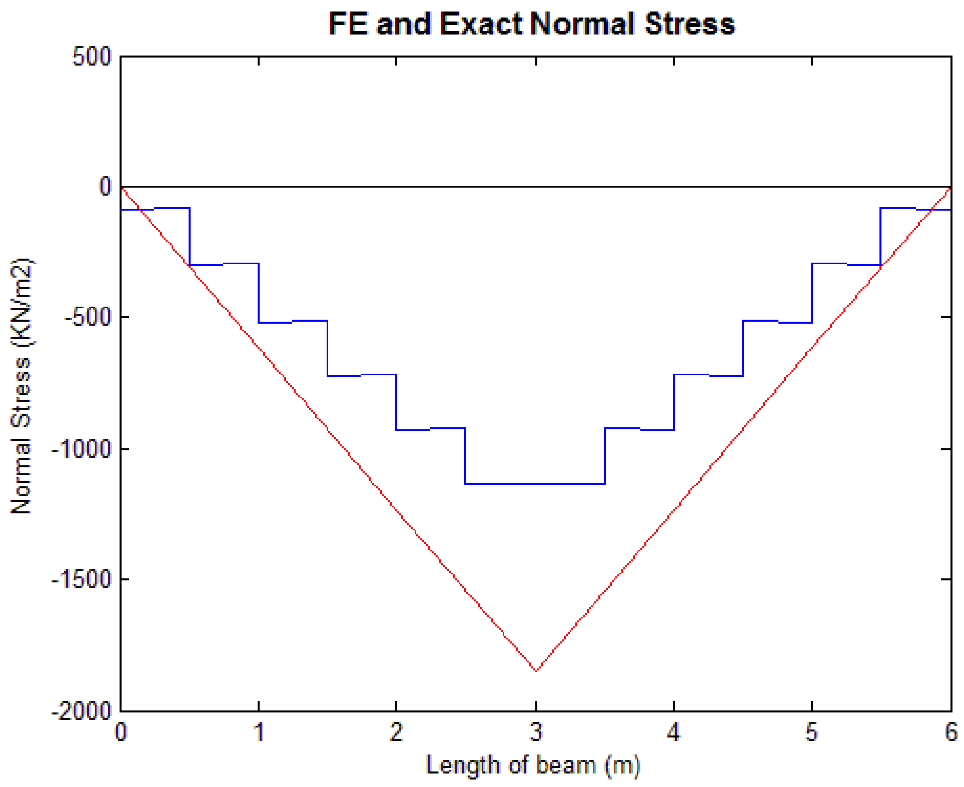
The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.5;   %Length of the element in x direction
dy = 0.225; %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
%Dividing the nodal forces along the elements on the left side
f(59*2,1) = -50/(ny+1);
iino = (nx/2)+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;  %k is the stiffness matrix
no = 53;
```

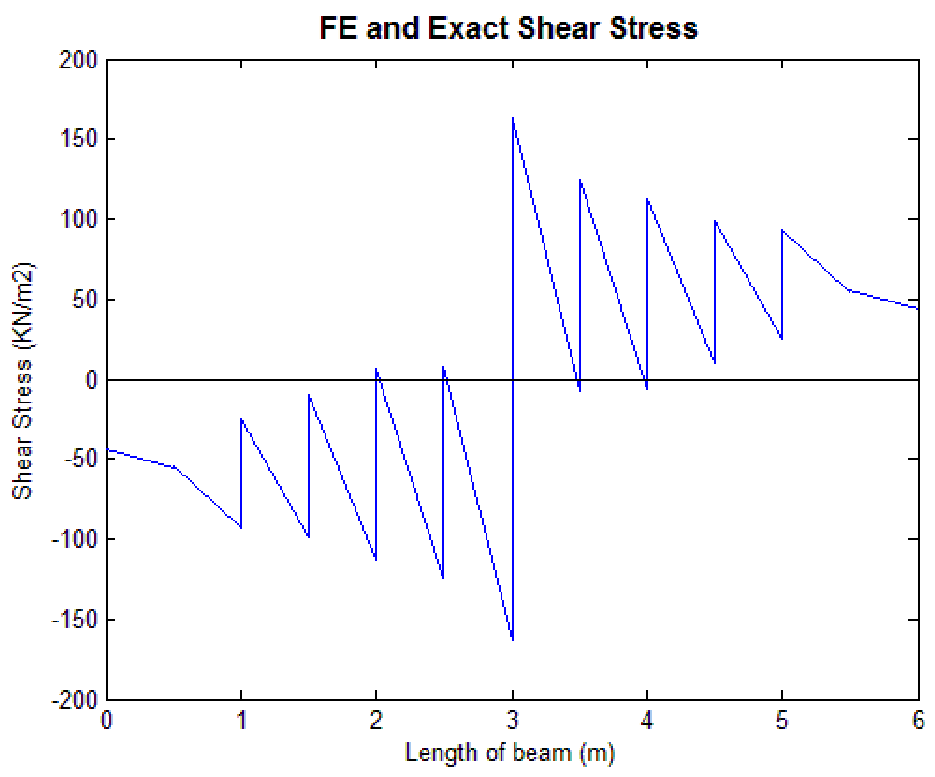
```
%Removing all the unknown forces and the corresponding rows and
%columns in the global stiffness matrix
f(26,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(26,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,26) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;26],u);   %U is the displacement vector
F = K*U;    %F is the force vector
```



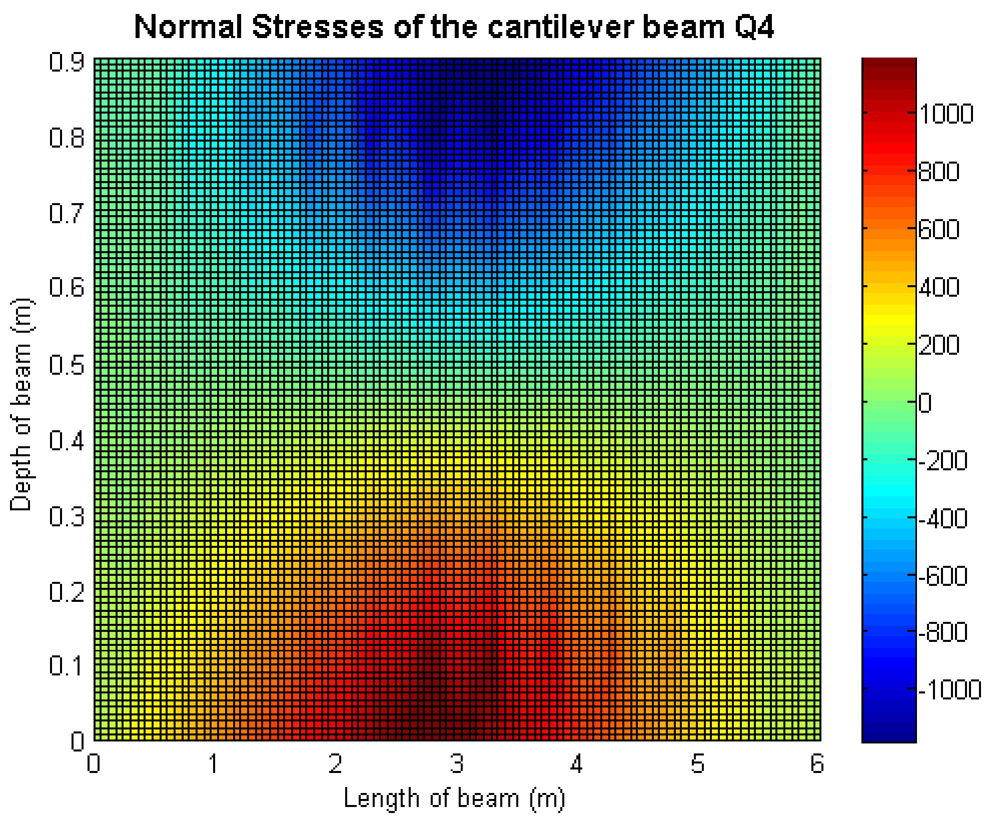
(a)



(b)



(b)



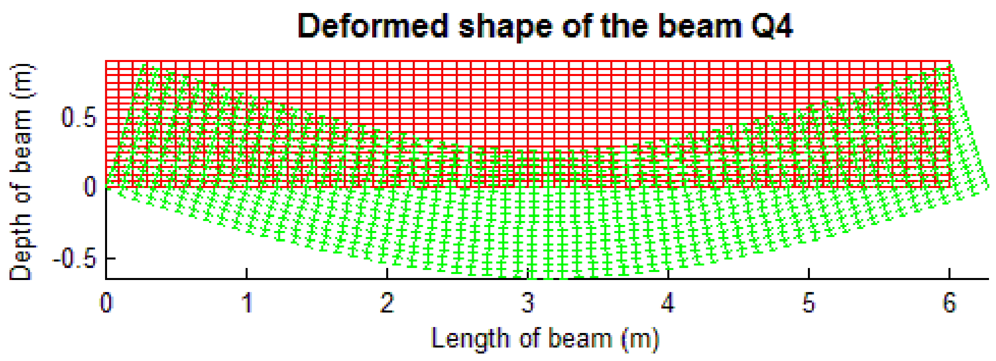
(c)

**Fig. 7-5** Analysis of CantileverBeam_C2_02_03_Q4

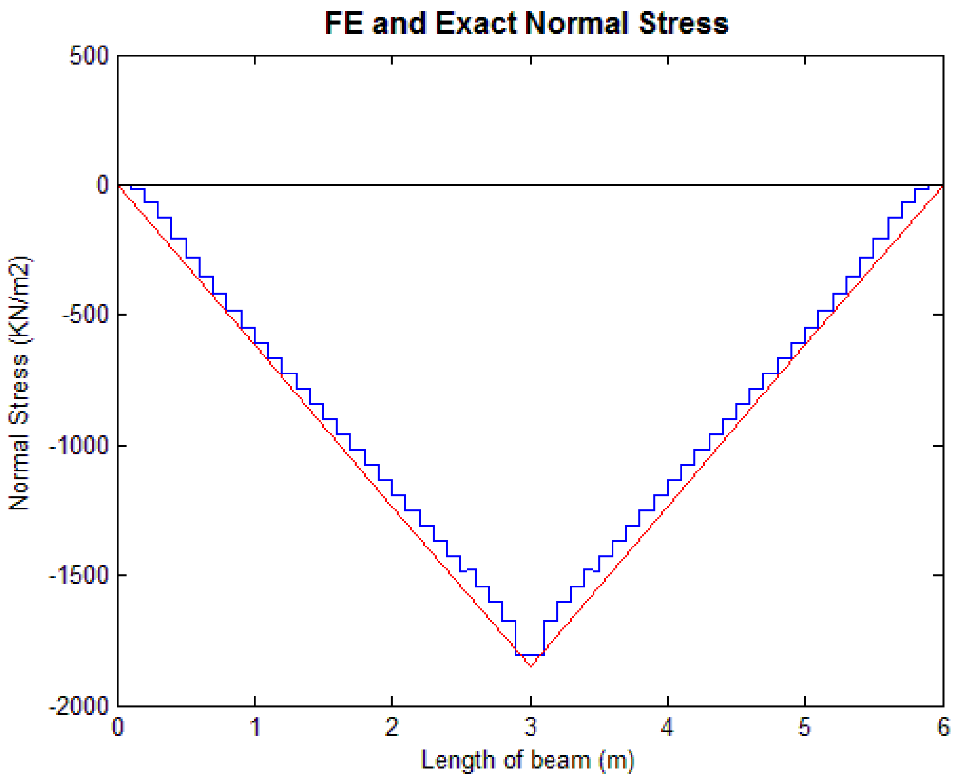
### *7.3.6 SimpleBeam_A1_03_03_Q4*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

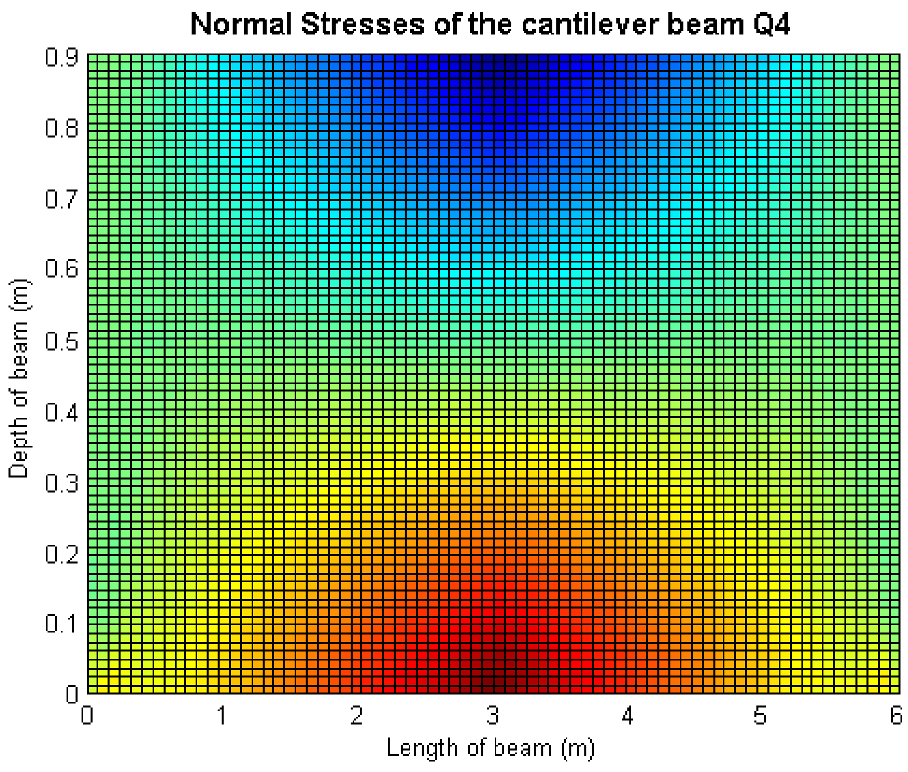
```
%Meshing
dx = 0.1;   %Length of the element in x direction
dy = 0.05;  %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
of the nodes and coordinates)
[element,K,centroid] =
BilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(2318,1);
ny = Ly/dy;
f(31*2,1) = -50/(ny+1);
iino = 31*2;
for ii = 1:ny;
    iino = iino+(61*2);
    f(iino) = -50/(ny+1);
end
f(122,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(122,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,122) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;122],u);
F = K*U;
```



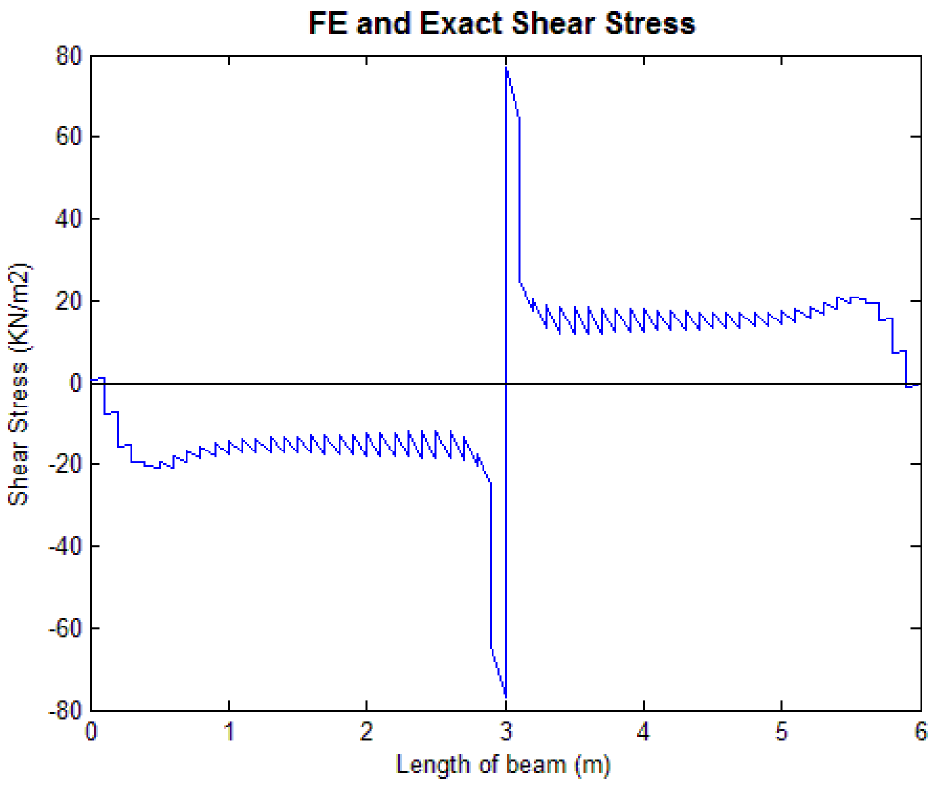
(a)


FE and Exact Normal Stress
500
0
-500
-1000
-1500
-2000
Normal Stress (KN/m2)
0
1
2
3
4
5
6
Length of beam (m)

(b)


Normal Stresses of the cantilever beam Q4
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Depth of beam (m)
0
1
2
3
4
5
6
Length of beam (m)

(c)



(d)

**Fig. 7-6** Analysis of SimpleBeam_A2_03_03_Q4

### 7.3.7 *Codes used in the script*

#### 7.3.7.1 *Bilinear Quadratic Element Stiffness*

```
function k =
BilinearQuadElementStiffness(E,NU,t,x1,y1,x2,y2,x3,y3,x4,y4,p)
%BilinearQuadElementStiffness This function returns the element
% stiffness matrix for a bilinear quadrilateral element with modulus
% of elasticity E, Poisson's ratio NU, thickness t, coordinates of
% node 1 (x1,y1), coordinates node 2 (x2,y2), coordinates node 3
%(x3,y3), and coordinates of node 4 (x4,y4). Use p = 1 for cases
% of plane stress, and p = 2 for cases of plane strain. The size of
%the element stiffness matrix is 8 x 8.
syms s t;
a = (y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s))/4;
b = (y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t))/4;
c = (x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t))/4;
d = (x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s))/4;
B1 = [a*(t-1)/4-b*(s-1)/4 0 ; 0 c*(s-1)/4-d*(t-1)/4
    c*(s-1)/4-d*(t-1)/4 a*(t-1)/4-b*(s-1)/4];
B2 = [a*(1-t)/4-b*(-1-s)/4 0 ; 0 c*(-1-s)/4-d*(1-t)/4
    c*(-1-s)/4-d*(1-t)/4 a*(1-t)/4-b*(-1-s)/4];
B3 = [a*(t+1)/4-b*(s+1)/4 0 ; 0 c*(s+1)/4-d*(t+1)/4
    c*(s+1)/4-d*(t+1)/4 a*(t+1)/4-b*(s+1)/4];
B4 = [a*(-1-t)/4-b*(1-s)/4 0 ; 0 c*(1-s)/4-d*(-1-t)/4
    c*(1-s)/4-d*(-1-t)/4 a*(-1-t)/4-b*(1-s)/4];
Bfirst = [B1 B2 B3 B4];
Js = [-(1-t); 1-t; 1+t; -(1+t)];
```

```
Jt = [-(1-s); -(1+s); 1+s; 1-s];
J1 = 1/16*[x1 x2 x3 x4]*Js*Jt'*[y1; y2; y3; y4];
J2 = 1/16*[x1 x2 x3 x4]*Jt*Js'*[y1; y2; y3; y4];
J = J1-J2;
B = Bfirst/J;
if p == 1
    D = (E/(1-NU*NU))*[1, NU, 0; NU, 1, 0; 0, 0, (1-NU)/2];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0; NU, 1-NU, 0;
    0, 0, (1-2*NU)/2];
end
BD = J*transpose(B)*D*B;
r = int(int(BD, t, -1, 1), s, -1, 1);
z = t*r;
k = double(z);
end
```

#### *7.3.7.2 Bilinear Quadratic Assemble*

```
function K = BilinearQuadAssemble(K,k,i,j,m,n)
%BilinearQuadAssemble This function assembles the element stiffness
%matrix k of the bilinear quadrilateral element with nodes i, j, m,
%and n into the global stiffness matrix K. This function returns the
%global stiffness matrix K after %the element stiffness matrix k is
%assembled.
K(2*i-1,2*i-1) = K(2*i-1,2*i-1)+k(1,1);
K(2*i-1,2*i) = K(2*i-1,2*i)+k(1,2);
K(2*i-1,2*j-1) = K(2*i-1,2*j-1)+k(1,3);
K(2*i-1,2*j) = K(2*i-1,2*j)+k(1,4);
K(2*i-1,2*m-1) = K(2*i-1,2*m-1)+k(1,5);
K(2*i-1,2*m) = K(2*i-1,2*m)+k(1,6);
K(2*i-1,2*n-1) = K(2*i-1,2*n-1)+k(1,7);
K(2*i-1,2*n) = K(2*i-1,2*n)+k(1,8);

K(2*i,2*i-1) = K(2*i,2*i-1)+k(2,1);
K(2*i,2*i) = K(2*i,2*i)+k(2,2);
K(2*i,2*j-1) = K(2*i,2*j-1)+k(2,3);
K(2*i,2*j) = K(2*i,2*j)+k(2,4);
K(2*i,2*m-1) = K(2*i,2*m-1)+k(2,5);
K(2*i,2*m) = K(2*i,2*m)+k(2,6);
K(2*i,2*n-1) = K(2*i,2*n-1)+k(2,7);
K(2*i,2*n) = K(2*i,2*n)+k(2,8);

K(2*j-1,2*i-1) = K(2*j-1,2*i-1)+k(3,1);
K(2*j-1,2*i) = K(2*j-1,2*i)+k(3,2);
K(2*j-1,2*j-1) = K(2*j-1,2*j-1)+k(3,3);
K(2*j-1,2*j) = K(2*j-1,2*j)+k(3,4);
K(2*j-1,2*m-1) = K(2*j-1,2*m-1)+k(3,5);
K(2*j-1,2*m) = K(2*j-1,2*m)+k(3,6);
K(2*j-1,2*n-1) = K(2*j-1,2*n-1)+k(3,7);
K(2*j-1,2*n) = K(2*j-1,2*n)+k(3,8);

K(2*j,2*i-1) = K(2*j,2*i-1)+k(4,1);
K(2*j,2*i) = K(2*j,2*i)+k(4,2);
K(2*j,2*j-1) = K(2*j,2*j-1)+k(4,3);
K(2*j,2*j) = K(2*j,2*j)+k(4,4);
K(2*j,2*m-1) = K(2*j,2*m-1)+k(4,5);
K(2*j,2*m) = K(2*j,2*m)+k(4,6);
K(2*j,2*n-1) = K(2*j,2*n-1)+k(4,7);
K(2*j,2*n) = K(2*j,2*n)+k(4,8);
```

```
K(2*m-1,2*i-1) = K(2*m-1,2*i-1)+k(5,1);
K(2*m-1,2*i) = K(2*m-1,2*i)+k(5,2);
K(2*m-1,2*j-1) = K(2*m-1,2*j-1)+k(5,3);
K(2*m-1,2*j) = K(2*m-1,2*j)+k(5,4);
K(2*m-1,2*m-1) = K(2*m-1,2*m-1)+k(5,5);
K(2*m-1,2*m) = K(2*m-1,2*m)+k(5,6);
K(2*m-1,2*n-1) = K(2*m-1,2*n-1)+k(5,7);
K(2*m-1,2*n) = K(2*m-1,2*n)+k(5,8);

K(2*m,2*i-1) = K(2*m,2*i-1)+k(6,1);
K(2*m,2*i) = K(2*m,2*i)+k(6,2);
K(2*m,2*j-1) = K(2*m,2*j-1)+k(6,3);
K(2*m,2*j) = K(2*m,2*j)+k(6,4);
K(2*m,2*m-1) = K(2*m,2*m-1)+k(6,5);
K(2*m,2*m) = K(2*m,2*m)+k(6,6);
K(2*m,2*n-1) = K(2*m,2*n-1)+k(6,7);
K(2*m,2*n) = K(2*m,2*n)+k(6,8);

K(2*n-1,2*i-1) = K(2*n-1,2*i-1)+k(7,1);
K(2*n-1,2*i) = K(2*n-1,2*i)+k(7,2);
K(2*n-1,2*j-1) = K(2*n-1,2*j-1)+k(7,3);
K(2*n-1,2*j) = K(2*n-1,2*j)+k(7,4);
K(2*n-1,2*m-1) = K(2*n-1,2*m-1)+k(7,5);
K(2*n-1,2*m) = K(2*n-1,2*m)+k(7,6);
K(2*n-1,2*n-1) = K(2*n-1,2*n-1)+k(7,7);
K(2*n-1,2*n) = K(2*n-1,2*n)+k(7,8);

K(2*n,2*i-1) = K(2*n,2*i-1)+k(8,1);
K(2*n,2*i) = K(2*n,2*i)+k(8,2);
K(2*n,2*j-1) = K(2*n,2*j-1)+k(8,3);
K(2*n,2*j) = K(2*n,2*j)+k(8,4);
K(2*n,2*m-1) = K(2*n,2*m-1)+k(8,5);
K(2*n,2*m) = K(2*n,2*m)+k(8,6);
K(2*n,2*n-1) = K(2*n,2*n-1)+k(8,7);
K(2*n,2*n) = K(2*n,2*n)+k(8,8);

end
```

### *7.3.7.3 Beam Meshing for Q4 element*

```
function [ element,nx,ny ] = BeamMeshingQ4( Lx,Ly,dx,dy )
%BeamMeshing   This function returns element matrix which contains of
%the elements, ID of nodes of each element and the coordinates of each
%node for each element. The size of element matrix is (number of
%elements*13). The inputs are the length of the beam Lx, its depth Ly,
%length of %the element in x direction dx and its length in y dy. This
%code is %done by certain algorithmic to have an automatic organized
%meshing for %a 4-noded element.

nx = Lx/dx; %number of elements in x direction
ny = Ly/dy; %number of elements in y direction
elements = nx*ny;   %total number of elements.

for r = 1:elements+ny
    a(r,1:4) = [r r+1 nx+2+r nx+1+r];
end
%matrix a has to do with the indices of the nodes
a(nx+1:nx+1:size(a,1),:) = [];
n = 0;
m = 1;
```

```
for c = 1:ny
    nn = 0;
    for cc = 1:nx+1
        b(m,1:8) = [nn*dx n*dy cc*dx n*dy cc*dx c*dy nn*dx c*dy];
        nn = 1+nn;
        m = m+1;
    end
    n = n+1;
end
%matrix b has to do with the corresponding positions for each node
b(nx+1:nx+1:size(b,1),:) = [];
for id = 1:elements
    idd(id,1) = id; %the index of the element
end
element=[idd a b];  %element matrix = [index of nodes, i, j, m, n, x1, 
y1, x2, y2, x3, y3, x4, y4]
end
```

#### 7.3.7.4 *Beam Meshing for Q4 element*

```
function [element,K,centroid] = 
BilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,p)
%BilinearQuadElementStiffnessAssembly This code returns the Global
%stiffness matrix for the whole structure to be analyzed.
%   The inputs are modulus of elasticity E, Poisson's ratio NU,
%thickness of the element t, length of the beam Lx, its depth Ly,
%length of element in x dx, length of the element in y Ly and p which
%is 1 in case of plain stress problems and 2 in case of plain strain
%problems. The output is the element matrix which contains all the
%details of each element for the beam, the whole global stiffness
%matrix for the whole beam K and the a matrix which contains the
%positions of the centroid of the elements centroid.
[element,nx,ny] = BeamMeshingQ4(Lx,Ly,dx,dy);

no = 0;
K = zeros(2*(nx+1)*(ny+1),2*(nx+1)*(ny+1));
for ii=1:size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    x1 = element(ii,6);
    y1 = element(ii,7);
    x2 = element(ii,8);
    y2 = element(ii,9);
    x3 = element(ii,10);
    y3 = element(ii,11);
    x4 = element(ii,12);
    y4 = element(ii,13);
    centroid(ii,:) = [(x2+x1)/2,(y3+y2)/2];
    k = 
BilinearQuadElementStiffness(E,NU,t,x1,y1,x2,y2,x3,y3,x4,y4,p);
    no = no + 1;
    K = BilinearQuadAssemble(K,k,i,j,m,n);
end
```

#### *7.3.7.5 Stress Calculation*

```
function [ sigma,sigmaX, sigmaBX, xB, yB, Qcentroid, Qtop ] =
StressCalculation( E,NU,U,Lx,Ly,dx,dy,element,p,xe,ye )
%StressDistribution     This function returns a graph between the
%stress values of the beam and its length. E is the modulus of
%elasticity, NU is Poisson's ratio, U is the displacement vector of
%the beam, Lx is the Length of the beam, Ly is the beam's height, dx
%is the length of the mesh in x-direction, dy is the length of the
%mesh in y-direction, element is a matrix contains information about
%the nodes and their coordinates, p is the problem that is going to
%be use (1 for stress and 2 for strain) and finally xe and ye are the
%coordinates of the point that the stresses are going to be
%calculates.

nx = Lx/dx;
ny = Ly/dy;
no = 0;
for q = 1: size(element,1)
    a = element(q,2);
    b = element(q,3);
    c = element(q,4);
    d = element(q,5);
    x1 = element(q,6);
    y1 = element(q,7);
    x2 = element(q,8);
    y2 = element(q,9);
    x3 = element(q,10);
    y3 = element(q,11);
    x4 = element(q,12);
    y4 = element(q,13);
    uu((8*no)+1:(8*no)+8,1) = [U((((a*2)-1)):(((a*2)+2)),1); U(((c*2)-
1),1);U((c*2),1);U(((d*2)-1),1);U((d*2),1)];

%sigma is the stresses at xe = [-dx/2 and dx/2] & ye = 0 to draw a
%distribution of the normal stresses along the beam and compare it to
%the exact ones
    sigma((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),xe,ye);
%to calculate shear stress
    sigmaQcentroid((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),xe,0);
%sigma0 is the stresses at the centroid of the element
    sigma0((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),0,0);
%sigmabottom is the stresses of all elements at their bottom side
    sigmabottom((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),-dx/2,-dy/2);
%sigmatop is the stesses of all elements at their top right side
    sigmatop((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),dx/2,dy/2);
%sigmaleft is the stresses of all elements at their top left side
    sigmaleft((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),-dx/2,dy/2);
%sigma right is the stresses of all elements at their bottom right
side
```

```
    sigmaright((3*no)+1:(3*no)+3,:) =
BilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((8*no)+1
:(8*no)+8,1),dx/2,-dy/2);
    no = no+1;
end

%n are the number of elements used
n = size(element,1);

%sigmaBX are the normal stresses at the boundary nodes of the beams
sigma0X = sigma0(1:3:size(sigma0),:);
sigmatopX = sigmatop(1:3:size(sigmatop),:);
sigmabottomX = sigmabottom(1:3:size(sigmabottom),:);
sigmaleftX = sigmaleft(1:3:size(sigmaleft),:);
sigmarightX = sigmaright(1:3:size(sigmaright),:);

x1 = element(:,6);
y1 = element(:,7);
x2 = element(:,8);
y2 = element(:,9);
x3 = element(:,10);
y3 = element(:,11);
x4 = element(:,12);
y4 = element(:,13);
sigmaBbottomX = sigmabottomX(1:nx,:);
xbottom = x1(1:nx,:);
ybottom = y1(1:nx,:);
sigmaBtopX = sigmatopX(n-nx+1:n,:);
xtop = x3(n-nx+1:n,:);
ytop = y3(n-nx+1:n,:);
sigmaBleftX = sigmaleftX(1:nx:n,:);
xleft = x4(1:nx:n,:);
yleft = y4(1:nx:n,:);
sigmaBrightX = sigmarightX(nx:nx:n,:);
xright = x2(nx:nx:n,:);
yright = y2(nx:nx:n,:);

sigmaBX = [sigma0X; sigmaBbottomX; sigmaBtopX; sigmaBleftX;
sigmaBrightX];
xB = [xbottom; xtop; xleft; xright];
yB = [ybottom; ytop; yleft; yright];

%sigmaX are the normal stresses
sigmaX = sigma(1:3:size(sigma),:);

%Q is the shear stress
Qcentroid = sigmaQcentroid(3:3:size(sigmaQcentroid),:);
Qtop = sigma(3:3:size(sigma),:);
end
```

#### 7.3.7.6 *Draw Stresses*

```
function [ sigmatop,sigmatop1 ] = DrawStresses( Lx,dx,element,stress )
%DrawStresses This function returns a graph for the normal stresses
along the beam. The inputs are length of beam Lx, length of element in
x direction, element matrix and stress vector.

nx = Lx/dx;
n = size(element,1);
```

```
%sigmatopX are the stresses starting from the left top element to the
right top one
sigmatop = stress(n-nx+1:n,:);

%Putting the stresses in one column instead of two to draw the
%shear/normal stress with the length of the beam
noo = 0;
for qq = 1:size(sigmatop)
    sigmatop1((2*noo)+1:(2*noo)+2,1) =
[sigmatop(qq,1);sigmatop(qq,2)];
    noo = noo+1;
end

% To create the length vector as an increment of the length of the
%element in x direction
L = 0;
for q = 1:nx
    L1(q,1) = L;
    L2(q,1) = L+dx;
    L=L+dx;
end

L1L2=[L1 L2];

nooo = 0;
for q = 1:size(L1L2)
    LL((2*nooo)+1:(2*nooo)+2,1) = [L1L2(q,1);L1L2(q,2)];
    nooo = nooo+1;
end
plot(LL,sigmatop1)

end
```

### *7.3.7.7 Draw Contour lines for normal stresses*

```
function [  ] = DrawContoursNormalStresses( sigmaBX, centroid, xB, yB,
Lx, Ly, no)
%DrawContoursNormalStresses     This function draws contour lines for
%the normal stresses on a beam where sigmaBX is the normal stresses
%in the centroid of the beam and the top and bottom edges of the top
%and bottom elements respectively, centroid is the coordinates of the
%centroid of the elements,xB and yB are the positions of the normal
%stresses at the top and bottom of the top and bottom elements, Lx is
%the length of the beam, Ly is its depth and no is the accuracy of
%the contour lines.
axis equal
a = [centroid(:,1); xB];
b = [centroid(:,2); yB];
c = sigmaBX(:,1);
alin = linspace(0,Lx,no);
blin = linspace(0,Ly,no);
[A,B] = meshgrid(alin,blin);
C = griddata(a,b,c,A,B,'natural');
surf(A,B,C)
xlabel('Length of beam (m)')
ylabel('Depth of beam (m)')
title('Normal Stresses of the cantilever beam
Q4','FontSize',12,'FontWeight','Bold')
view(2)
end
```

#### *7.3.7.8 Draw Beam*

```
function y = DrawBeam( element )
%DrawBeam This function returns a graph for a beam after meshing with
%4-node elements.
%   The input is element matrix

hold on
axis equal
for ii = 1: size(element,1)
    x1 = element(ii,6);
    y1 = element(ii,7);
    x2 = element(ii,8);
    y2 = element(ii,9);
    x3 = element(ii,10);
    y3 = element(ii,11);
    x4 = element(ii,12);
    y4 = element(ii,13);
    plot([x1, x2], [y1, y2],'color',[1 0 0]);
    plot([x2, x3], [y2, y3],'color',[1 0 0]);
    plot([x3, x4], [y3, y4],'color',[1 0 0]);
    plot([x4, x1], [y4, y1],'color',[1 0 0]);
end
end
```

#### *7.3.7.9 Draw Deformed Beam*

```
function [Ae,Aenew ] = DrawDeformedBeam( element,U )
%DrawDeformedBeam This function returns a graph for the beam after
%deformation. The input is element matrix and Displacement vector U.
(It is better to multiply U vector by certain value for better graph.

hold on
axis equal
for ii = 1: size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    x1 = element(ii,6);
    y1 = element(ii,7);
    x2 = element(ii,8);
    y2 = element(ii,9);
    x3 = element(ii,10);
    y3 = element(ii,11);
    x4 = element(ii,12);
    y4 = element(ii,13);
    ux1 = U((i*2)-1,1);
    ux2 = U((j*2)-1,1);
    ux3 = U((m*2)-1,1);
    ux4 = U((n*2)-1,1);
    uy1 = U((i*2),1);
    uy2 = U((j*2),1);
    uy3 = U((m*2),1);
    uy4 = U((n*2),1);
    x1new = x1+ux1;
    x2new = x2+ux2;
    x3new = x3+ux3;
    x4new = x4+ux4;
    y1new = y1+uy1;
    y2new = y2+uy2;
    y3new = y3+uy3;
    y4new = y4+uy4;
    centroidnew(ii,:) = [(((x2new+x1new)/2)+((x3new...
        +x4new)/2))/2,(((y3new+y2new)/2)+((y1new...
        +y4new)/2))/2];
    plot([x1new, x2new], [y1new, y2new],':','color',[0 1 0]);
    plot([x2new, x3new], [y2new, y3new],':','color',[0 1 0]);
    plot([x3new, x4new], [y3new, y4new],':','color',[0 1 0]);
    plot([x4new, x1new], [y4new, y1new],':','color',[0 1 0]);
    Ae(ii,1) = BilinearQuadElementArea(x1,y1,x2,y2,x3,y3,x4,y4);
    Aenew(ii,1) =
BilinearQuadElementArea(x1new,y1new,x2new,y2new,x3new,...
        y3new,x4new,y4new);
end
axis tight
xlabel('Length of beam (m)')
ylabel('Depth of beam (m)')
title('Deformed shape of the beam
Q4','FontSize',12,'FontWeight','Bold')
hold off
end
```

## 7.4 Appendix D: Q8 element

### *7.4.1 Cantilever_C2_01_03_Q8*

```
%...................................................................
%Cantilever_C1_01_03_Q8
%This script calculates the normal and shear stresses of a cantilever
%beam loaded by 1 concentrated load at its edge. This beam is modeled
%by Q8 element.
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
f(28,1) = -25;
f(14,1) = -25;
f(16,:) = [];
f(15,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(16,:) = [];
k(15,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,16) = [];
k(:,15) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([16;15;2;1],u);
F = K*U;

%Calculation of the stresses sigma
[sigma,sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculationQ8(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);

%Draw the normal stress
```

```
figure(2)
DrawStresses(Lx, dx, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeamQ8(element)
DrawDeformedBeamQ8(element,U*1000);
hold off

%Exact analysis
CantileverBeam_C1
%......................................................................
```

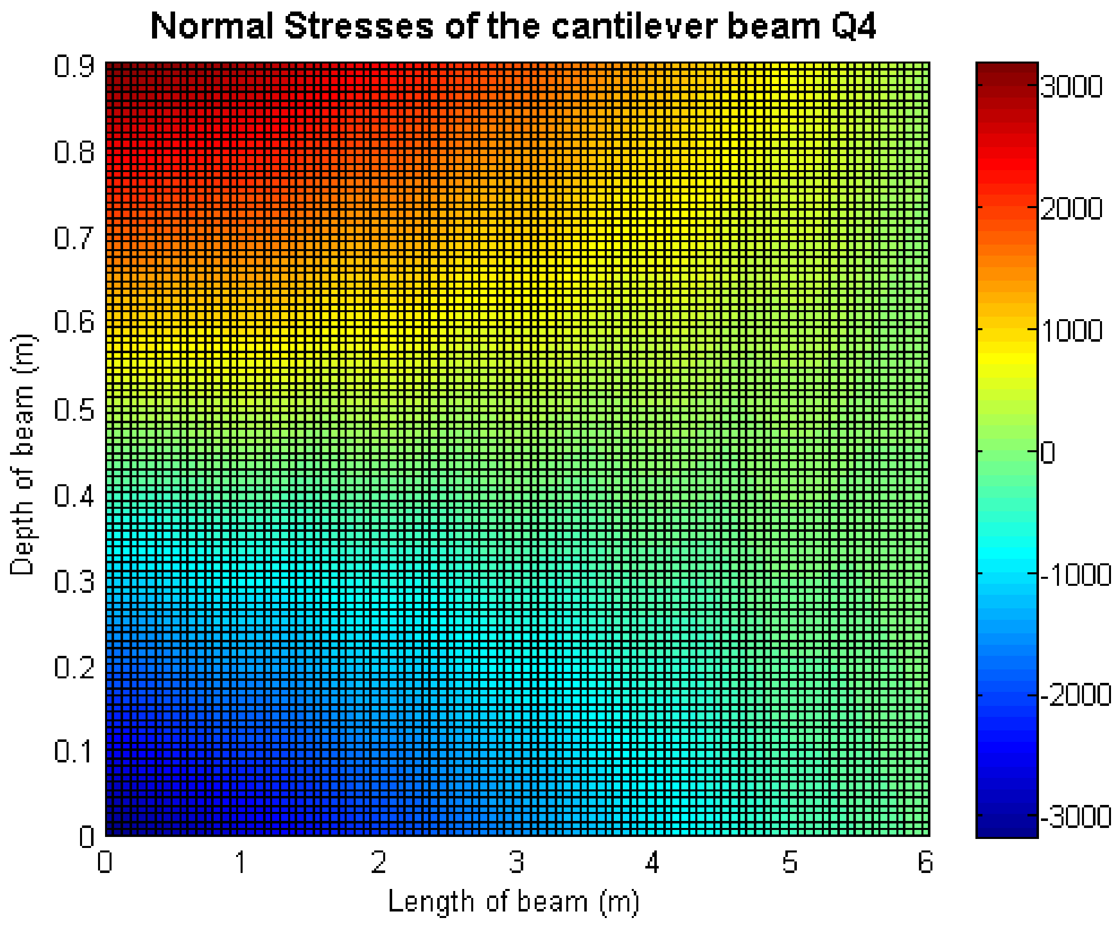


**Fig. 7-7** Normal Stresses of CantileverBeam_C2_01_03_Q8

### *7.4.2 Cantilever_C2_02_03_Q8*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.5; %Length of the element in x direction
dy = 0.225;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K,1),1);
nx = Lx/dx;
ny = Ly/dy;
f(65*2,1) = -50/(ny+1);
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 53;
nooo = 153;
for ii = 1:ny
    f(nooo*2,:) = [];
    f((nooo*2)-1,:) = [];
    k(nooo*2,:) = [];
    k((nooo*2)-1,:) = [];
    k(:,nooo*2) = [];
    k(:,(nooo*2)-1)=[];
    noooo((ii*2),1) = nooo*2;
    noooo((ii*2)-1,1) = (nooo*2)-1;
    nooo = nooo - 2*nx - 1;
end
for i = 1: ny+1
    f(no*2,:) = [];
    f((no*2)-1,:) = [];
    k(no*2,:) = [];
    k((no*2)-1,:) = [];
    k(:,no*2) = [];
    k(:,(no*2)-1)=[];
    noo((i*2),1) = no*2;
    noo((i*2)-1,1) = (no*2)-1;
    no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement([noo;noooo],u);
F = K*U;
```

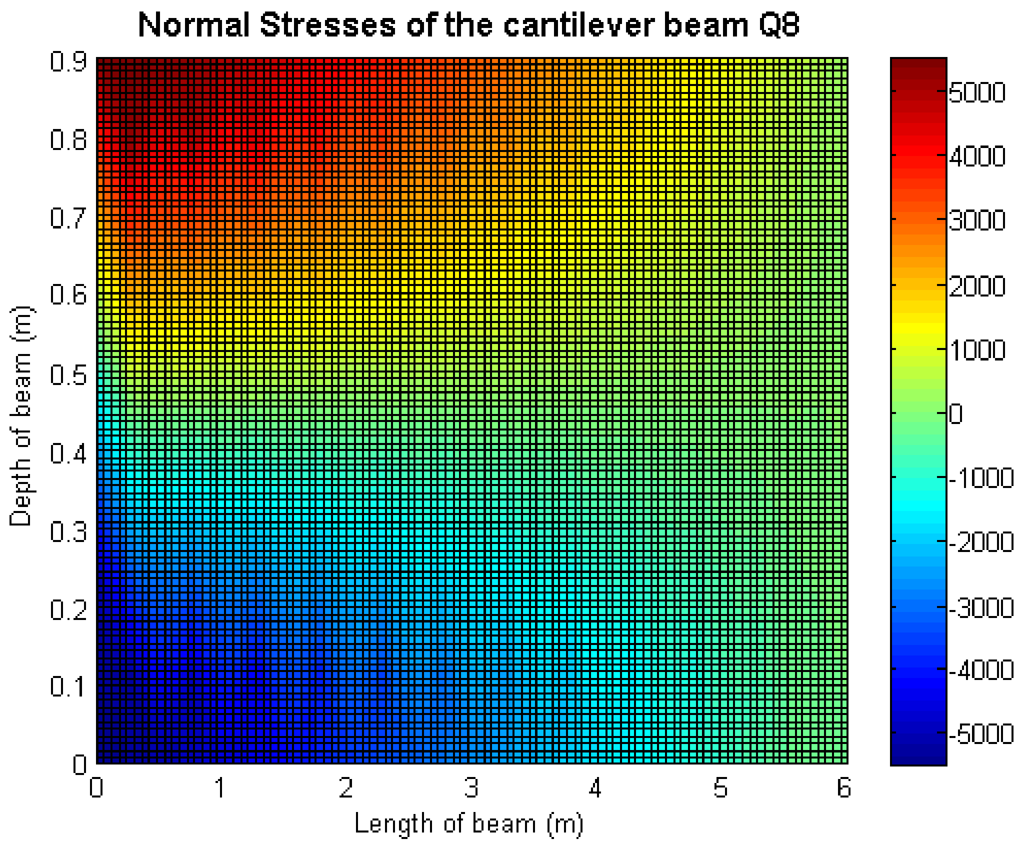


**Fig. 7-8** Normal Stresses of CantileverBeam_C2_02_03_Q8

### 7.4.3 *Cantilever_C2_03_03_Q8*

```
%.....................................................................
```

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.1; %Length of the element in x direction
dy = 0.05;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K,1),1);
nx = Lx/dx;
ny = Ly/dy;
f(1159*2,1)  = -50/(ny+1);
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1)  = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 1099;
```

```
nooo = 3277;
for ii = 1:ny
    f(nooo*2,:) = [];
    f((nooo*2)-1,:) = [];
    k(nooo*2,:) = [];
    k((nooo*2)-1,:) = [];
    k(:,nooo*2) = [];
    k(:,(nooo*2)-1)=[];
    noooo((ii*2),1) = nooo*2;
    noooo((ii*2)-1,1) = (nooo*2)-1;
    nooo = nooo - 2*nx - 1;
end
for i = 1: ny+1
    f(no*2,:) = [];
    f((no*2)-1,:) = [];
    k(no*2,:) = [];
    k((no*2)-1,:) = [];
    k(:,no*2) = [];
    k(:,(no*2)-1)=[];
    noo((i*2),1) = no*2;
    noo((i*2)-1,1) = (no*2)-1;
    no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement([noo;noooo],u);
F = K*U;
```

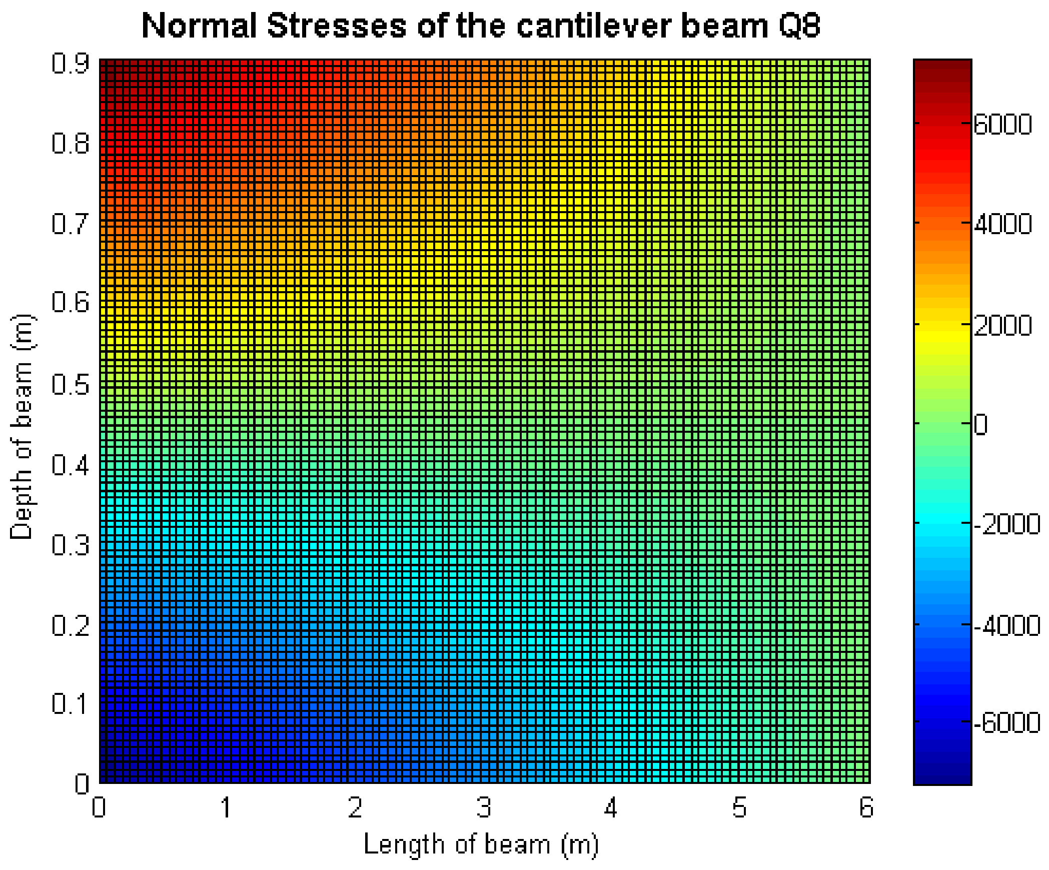


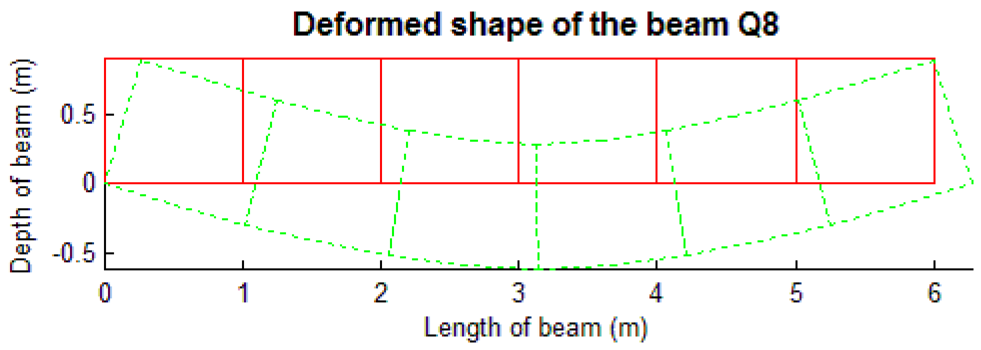
**Fig. 7-9** Normal Stresses of CantileverBeam_C2_03_03_Q8

### 7.4.4 *SimpleBeam_C2_01_03_Q8*

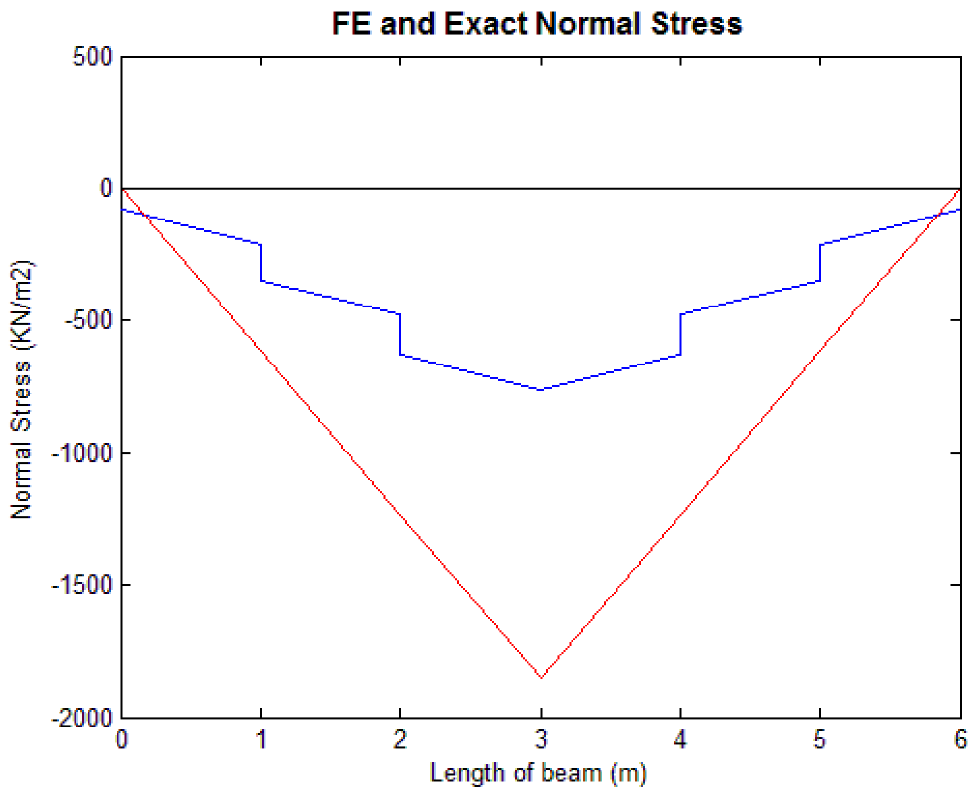
```
%.....................................................................
```

```
%SimpleBeam_A2_01_03_Q8
%This script calculates the normal and shear stresses of a simple beam
%by 1 concentrated load at mid-span. This beam is modeled by Q8
%element.
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
%numbering of the nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
f(22,1) = -25;
f(8,1) = -25;
f(14,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;  %k is the stiffness matrix
k(14,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,14) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([14;2;1],u);  %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (?x, ?y and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses ?x at the boundary nodes of the beam
%xB & yB are the positions of the sigmaBX stresses
%Qc is the shear stresses at the centroid of the beam
%Q0 is the min shear stresses which is at the top elements of the beam
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculationQ8(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);

%Draw the normal stress
figure(2)
DrawStresses(Lx, Ly, dx, dy, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, Ly, dx, dy, element, Qc);
%Contours for normal stress
figure(5)
```

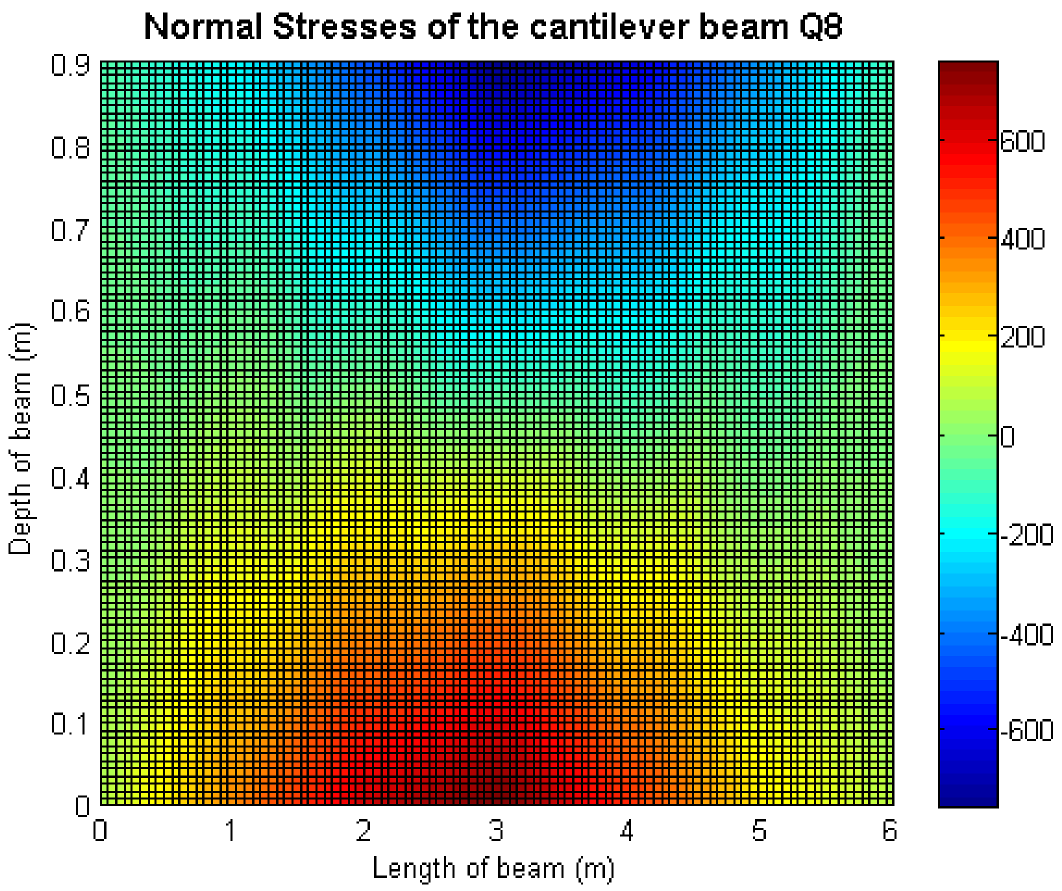
```
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeamQ8(element)
DrawDeformedBeamQ8(element,U*10000);
hold off

%Exact analysis
SimpleBeam_A1
%.....................................................................
```



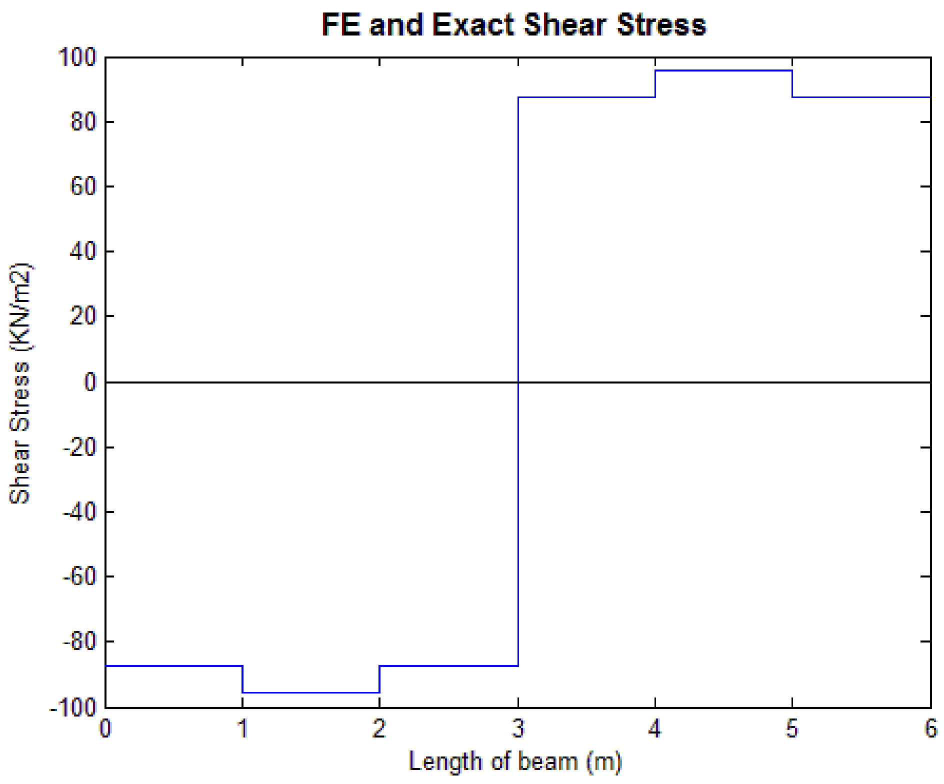
(a)



(b)



(c)



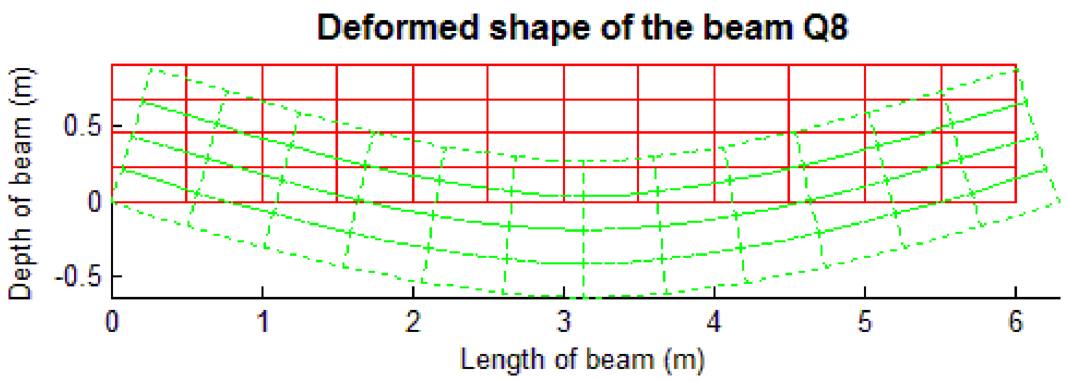
(d)

**Fig. 7-10** Analysis of SimpleBeam_A2_01_03_Q4

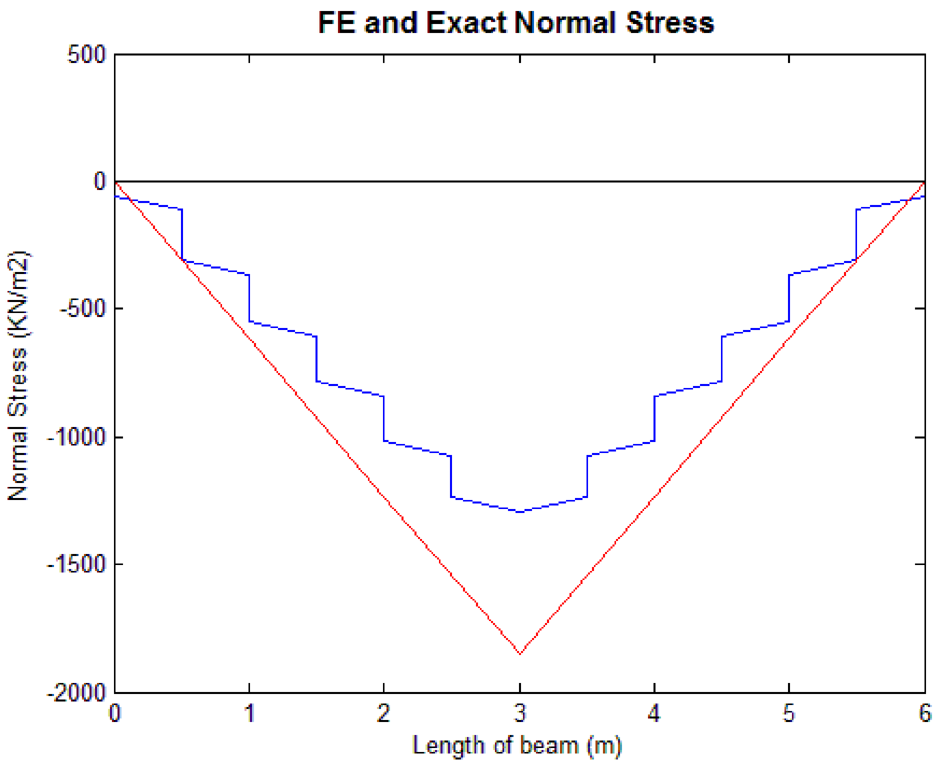
### 7.4.5 *SimpleBeam_C2_02_03_Q8*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

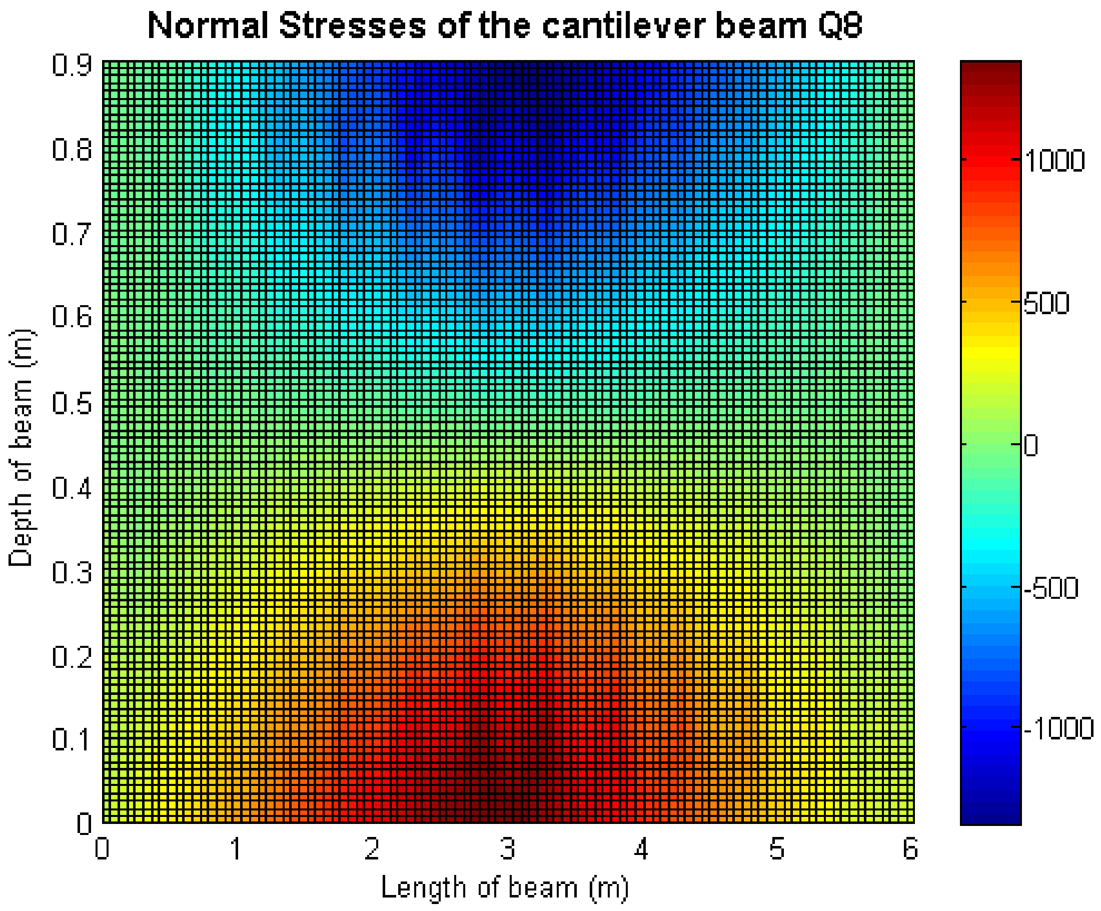
```
%Meshing
dx = 0.5;   %Length of the element in x direction
dy = 0.225; %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
of the
%nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
%Dividing the nodal forces along the elements on the left side
f(59*2,1) = -50/(ny+1);
iino = (nx/2)+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;  %k is the stiffness matrix
no = 53;
%Removing all the unknown forces and the corresponding rows and
%columns in the global stiffness matrix
f(26,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(26,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,26) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;26],u);   %U is the displacement vector
F = K*U;    %F is the force vector
```



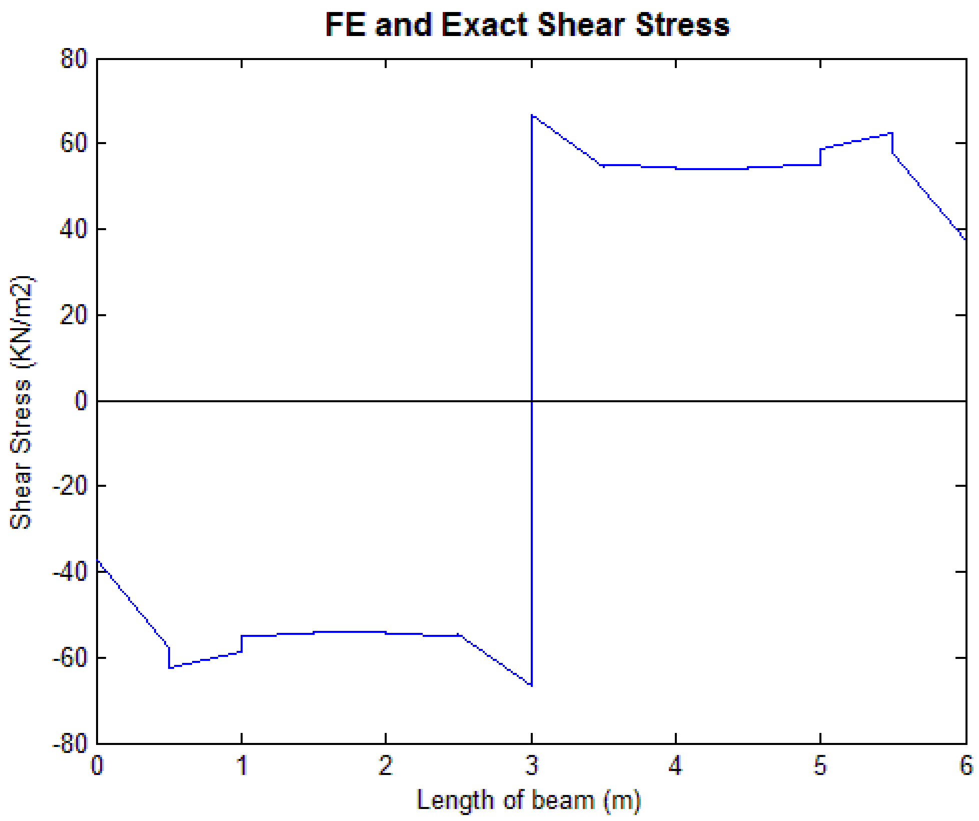
(a)


FE and Exact Normal Stress
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6
Normal Stress (KN/m2)
Length of beam (m)

(b)


Normal Stresses of the cantilever beam Q8
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
Depth of beam (m)
Length of beam (m)
1000
500
0
-500
-1000

(c)



(d)

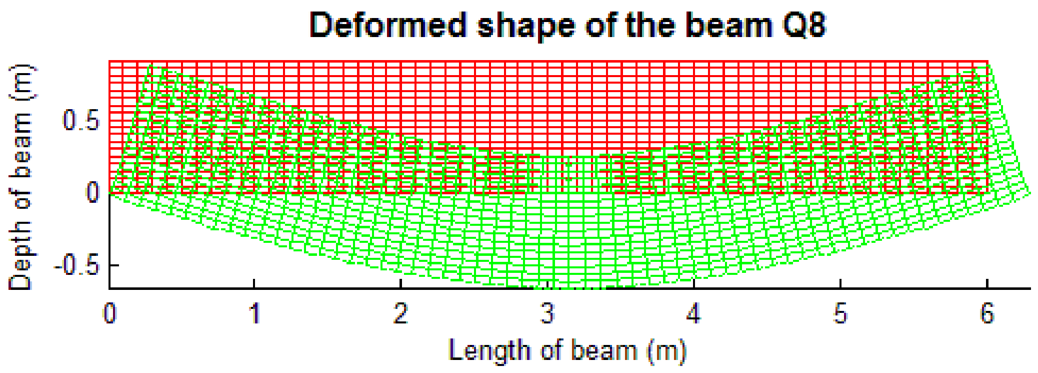
**Fig.7-11** Analysis of SimpleBeam_A2_02_03_Q4

### *7.4.6 SimpleBeam_C2_03_03_Q8*

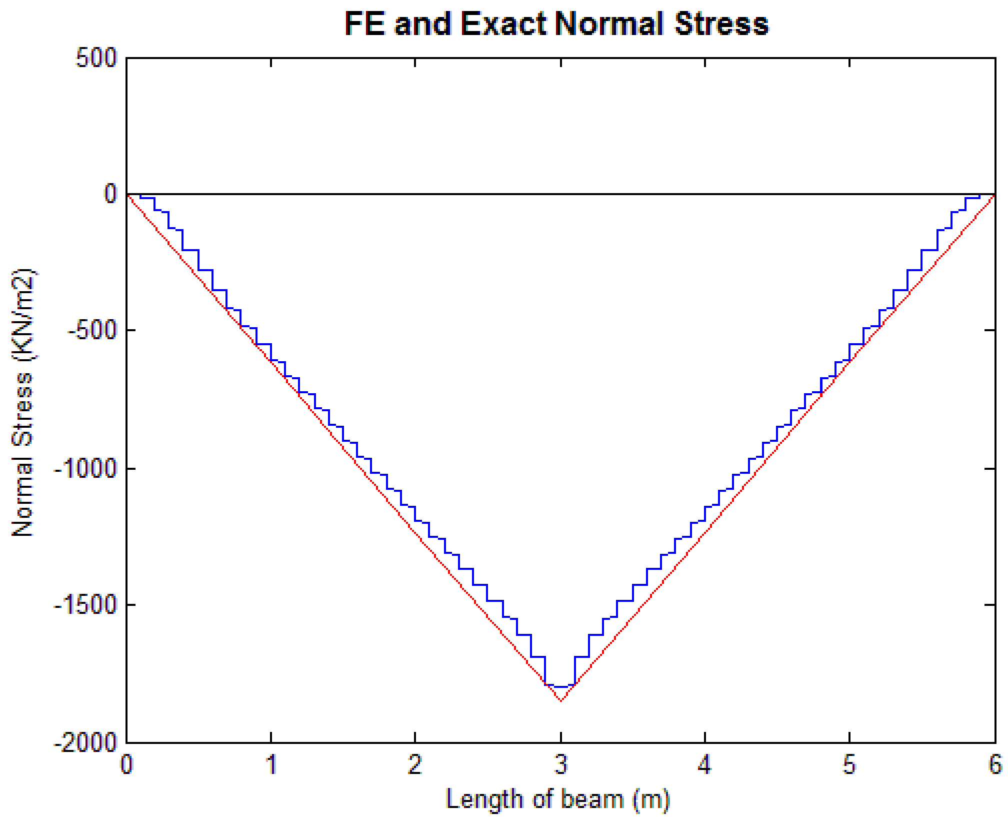
The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.1;    %Length of the element in x direction
dy = 0.05;   %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
of the
%nodes and coordinates)
[element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(2318,1);
ny = Ly/dy;
f(31*2,1) = -50/(ny+1);
iino = 31*2;
for ii = 1:ny;
    iino = iino+(61*2);
    f(iino) = -50/(ny+1);
end
f(122,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
```

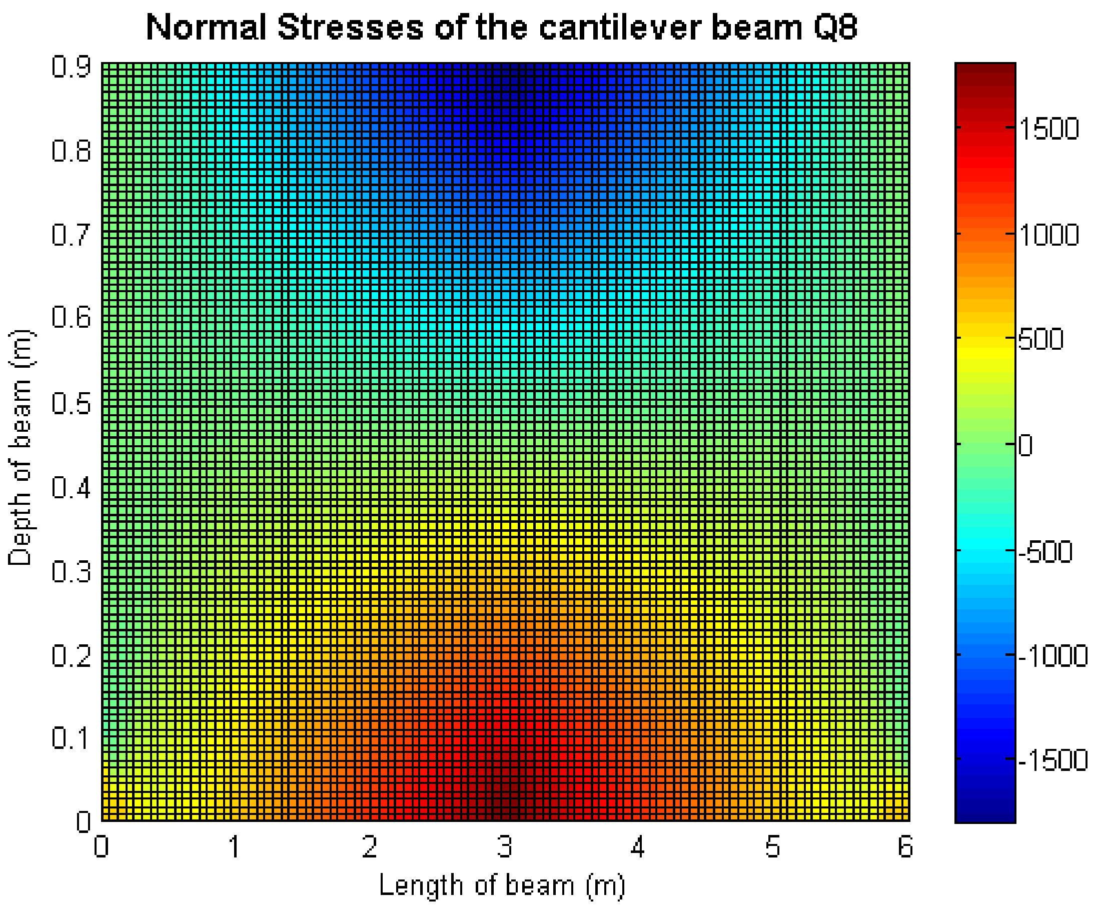
```
k(122,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,122) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;122],u);
F = K*U;
```



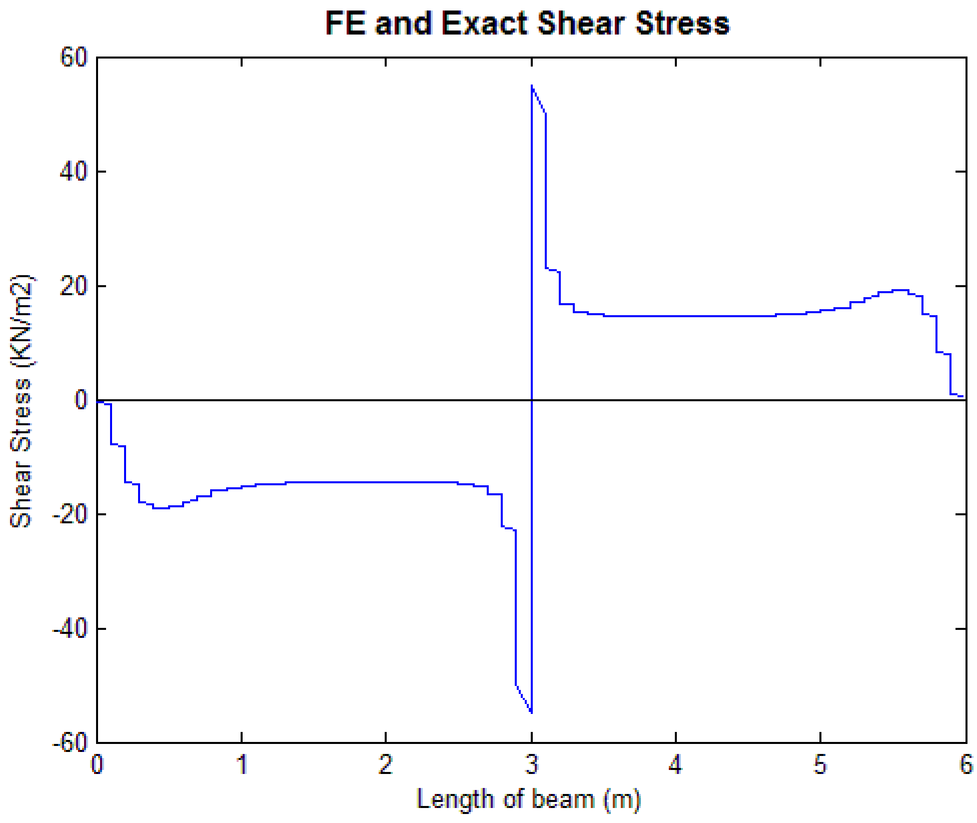
(a)



(b)



(c)



(d)

**Fig. 7-12** Analysis of SimpleBeam_A2_03_03_Q4

### 7.4.7 *Other codes*

#### 7.4.7.1 *Quadratic Quadrature Element Stiffness Assembly*

```
function [element,K,centroid] =
QuadraticQuadElementStiffnessAssembly(E,NU,h,Lx,Ly,dx,dy,p)
%QuadraticQuadElementStiffnessAssembly This code returns the Global
%stiffness matrix for a beam/plate to be analyzed as 8-noded element.
%   The inputs are modulus of elasticity E, Poisson's ratio NU,
%thickness of the element t, length of the beam Lx, its depth Ly,
%length of element in x dx, length of the element in y Ly and p which
%is 1 in case of plain stress problems and 2 in case of plain strain
%problems. The output is the element matrix which contains all the
%details of each element for the beam, the whole global stiffness
%matrix for the whole beam K and the a matrix which contains the
%positions of the centroid of the elements centroid.

[element,nx,ny] = BeamMeshingQ8(Lx,Ly,dx,dy);


no = 0;
K =
zeros((2*(nx+1)*(ny+1))+2*(nx*(ny+1))+2*(ny*(nx+1)),2*(nx+1)*(ny+1)+2*
(nx*(ny+1))+2*(ny*(nx+1)));
for ii=1:size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    ij = element(ii,6);
    jm = element(ii,7);
    mn = element(ii,8);
    ni = element(ii,9);
    x1 = element(ii,10);
    y1 = element(ii,11);
    x2 = element(ii,12);
    y2 = element(ii,13);
    x3 = element(ii,14);
    y3 = element(ii,15);
    x4 = element(ii,16);
    y4 = element(ii,17);
    centroid(ii,:) = [(x2+x1)/2,(y3+y2)/2];
    k =
QuadraticQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,y4,p);
    mat((16*no)+1:(16*no)+16,1:16) = k;
    no = no + 1;
   K = QuadraticQuadAssemble(K,k,i,j,m,n,ij,jm,mn,ni);
end
```

#### 7.4.7.2 *Beam Meshing Q8*

```
function [ element,nx,ny ] = BeamMeshingQ8( Lx,Ly,dx,dy )
%BeamMeshingQ8  This function returns element matrix which contains ID
%of elements, ID of nodes of each element and the coordinates
%of each node for each element. element matrix is (number of
%elements*17.
nx = Lx/dx;
ny = Ly/dy;
elements = nx*ny;
```

```
for r = 1:elements+ny
    a(r,1:4) = [r r+1 nx+2+r nx+1+r];
end

rno = elements+nx+ny+1;
for r = 1:elements+nx*(ny-1)+(ny-1)
    d(r,1:4) = [rno+r rno+r+nx+1 rno+r+2*nx+1 rno+r+nx];
end
a(nx+1:nx+1:size(a,1),:) = [];

for rr = nx+1: nx: size(a,1)
    for rrr = nx+1: 2*nx+1
        d(rr,:) = [];
    end
end

n = 0;
m = 1;
for c = 1:ny
    nn = 0;
    for cc = 1:nx+1
        b(m,1:8) = [nn*dx n*dy cc*dx n*dy cc*dx c*dy nn*dx c*dy];
        nn = 1+nn;
        m = m+1;
    end
    n = n+1;
end
b(nx+1:nx+1:size(b,1),:) = [];
for id = 1:elements
    idd(id,1) = id;
end
element=[idd a d b];

end
```

### 7.4.7.3 *Quadratic Quadrature Element Stiffness Assembly*

```
function w =
QuadraticQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,y4,p)
%QuadraticQuadElementStiffness This function returns the element
% stiffness matrix for a quadratic quadrilateral element with modulus
% of elasticity E, Poisson's ratio NU, thickness h, coordinates of
% node 1 (x1,y1), coordinates of node 2 (x2,y2), coordinates of
% node 3 (x3,y3), and coordinates of node 4 (x4,y4). Use p = 1 for
%cases of plane stress, and p = 2 for cases of plane strain.
syms s t;
x5 = (x1 + x2)/2;
x6 = (x2 + x3)/2;
x7 = (x3 + x4)/2;
x8 = (x4 + x1)/2;
y5 = (y1 + y2)/2;
y6 = (y2 + y3)/2;
y7 = (y3 + y4)/2;
y8 = (y4 + y1)/2;
N1 = (1-s)*(1-t)*(-s-t-1)/4;
N2 = (1+s)*(1-t)*(s-t-1)/4;
N3 = (1+s)*(1+t)*(s+t-1)/4;
N4 = (1-s)*(1+t)*(-s+t-1)/4;
N5 = (1-t)*(1+s)*(1-s)/2;
N6 = (1+s)*(1+t)*(1-t)/2;
N7 = (1+t)*(1+s)*(1-s)/2;
```

```
N8 = (1-s)*(1+t)*(1-t)/2;
x = N1*x1 + N2*x2 + N3*x3 + N4*x4 + N5*x5 + N6*x6 + N7*x7 + N8*x8;
y = N1*y1 + N2*y2 + N3*y3 + N4*y4 + N5*y5 + N6*y6 + N7*y7 + N8*y8;
xs = diff(x,s);
xt = diff(x,t);
ys = diff(y,s);
yt = diff(y,t);
J = xs*yt - ys*xt;
N1s = diff(N1,s);
N2s = diff(N2,s);
N3s = diff(N3,s);
N4s = diff(N4,s);
N5s = diff(N5,s);
N6s = diff(N6,s);
N7s = diff(N7,s);
N8s = diff(N8,s);
N1t = diff(N1,t);
N2t = diff(N2,t);
N3t = diff(N3,t);
N4t = diff(N4,t);
N5t = diff(N5,t);
N6t = diff(N6,t);
N7t = diff(N7,t);
N8t = diff(N8,t);
B11 = yt*N1s - ys*N1t;
B12 = 0;
B13 = yt*N2s - ys*N2t;
B14 = 0;
B15 = yt*N3s - ys*N3t;
B16 = 0;
B17 = yt*N4s - ys*N4t;
B18 = 0;
B19 = yt*N5s - ys*N5t;
B110 = 0;
B111 = yt*N6s - ys*N6t;
B112 = 0;
B113 = yt*N7s - ys*N7t;
B114 = 0;
B115 = yt*N8s - ys*N8t;
B116 = 0;
B21 = 0;
B22 = xs*N1t - xt*N1s;
B23 = 0;
B24 = xs*N2t - xt*N2s;
B25 = 0;
B26 = xs*N3t - xt*N3s;
B27 = 0;
B28 = xs*N4t - xt*N4s;
B29 = 0;
B210 = xs*N5t - xt*N5s;
B211 = 0;
B212 = xs*N6t - xt*N6s;
B213 = 0;
B214 = xs*N7t - xt*N7s;
B215 = 0;
B216 = xs*N8t - xt*N8s;
B31 = xs*N1t - xt*N1s;
B32 = yt*N1s - ys*N1t;
B33 = xs*N2t - xt*N2s;
B34 = yt*N2s - ys*N2t;
B35 = xs*N3t - xt*N3s;
B36 = yt*N3s - ys*N3t;
B37 = xs*N4t - xt*N4s;
B38 = yt*N4s - ys*N4t;
```

```
B39 = xs*N5t - xt*N5s;
B310 = yt*N5s - ys*N5t;
B311 = xs*N6t - xt*N6s;
B312 = yt*N6s - ys*N6t;
B313 = xs*N7t - xt*N7s;
B314 = yt*N7s - ys*N7t;
B315 = xs*N8t - xt*N8s;
B316 = yt*N8s - ys*N8t;
B = [B11 B12 B13 B14 B15 B16 B17 B18 B19 B110 B111 B112 B113 B114 B115
B116;B21 B22 B23 B24 B25 B26 B27 B28 B29 B210 B211 B212 B213 B214 B215
B216; B31 B32 B33 B34 B35 B36 B37 B38 B39 B310 B311 B312 B313 B314
B315 B316];
if p == 1
D = (E/(1-NU*NU))*[1, NU, 0 ; NU, 1, 0 ; 0, 0, (1-NU)/2];
elseif p == 2
D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0 ;NU, 1-NU, 0 ; 0, 0, (1-2*NU)/2];
end
Bnew = simplify(B);
Jnew = simplify(J);
BD = transpose(Bnew)*D*Bnew/Jnew;
r = int(int(BD, t, -1, 1), s, -1, 1);
z = h*r;
w = double(z);
end
```

### *7.4.7.4 Quadratic Quadrature Assemble*

```
function y = QuadraticQuadAssemble(K,k,i,j,m,p,q,r,s,t)
%QuadraticQuadAssemble This function assembles the element
% stiffness matrix k of the quadratic quadrilateral element with nodes
% i, j, m, p, q, r, s, and t into the global stiffness matrix K.
% This function returns the global stiffness matrix K after the
%element stiffness matrix k is assembled.
K(2*i-1,2*i-1) = K(2*i-1,2*i-1) + k(1,1);
K(2*i-1,2*i) = K(2*i-1,2*i) + k(1,2);
K(2*i-1,2*j-1) = K(2*i-1,2*j-1) + k(1,3);
K(2*i-1,2*j) = K(2*i-1,2*j) + k(1,4);
K(2*i-1,2*m-1) = K(2*i-1,2*m-1) + k(1,5);
K(2*i-1,2*m) = K(2*i-1,2*m) + k(1,6);
K(2*i-1,2*p-1) = K(2*i-1,2*p-1) + k(1,7);
K(2*i-1,2*p) = K(2*i-1,2*p) + k(1,8);
K(2*i-1,2*q-1) = K(2*i-1,2*q-1) + k(1,9);
K(2*i-1,2*q) = K(2*i-1,2*q) + k(1,10);
K(2*i-1,2*r-1) = K(2*i-1,2*r-1) + k(1,11);
K(2*i-1,2*r) = K(2*i-1,2*r) + k(1,12);
K(2*i-1,2*s-1) = K(2*i-1,2*s-1) + k(1,13);
K(2*i-1,2*s) = K(2*i-1,2*s) + k(1,14);
K(2*i-1,2*t-1) = K(2*i-1,2*t-1) + k(1,15);
K(2*i-1,2*t) = K(2*i-1,2*t) + k(1,16);
K(2*i,2*i-1) = K(2*i,2*i-1) + k(2,1);
K(2*i,2*i) = K(2*i,2*i) + k(2,2);
K(2*i,2*j-1) = K(2*i,2*j-1) + k(2,3);
K(2*i,2*j) = K(2*i,2*j) + k(2,4);
K(2*i,2*m-1) = K(2*i,2*m-1) + k(2,5);
K(2*i,2*m) = K(2*i,2*m) + k(2,6);
K(2*i,2*p-1) = K(2*i,2*p-1) + k(2,7);
K(2*i,2*p) = K(2*i,2*p) + k(2,8);
K(2*i,2*q-1) = K(2*i,2*q-1) + k(2,9);
K(2*i,2*q) = K(2*i,2*q) + k(2,10);
K(2*i,2*r-1) = K(2*i,2*r-1) + k(2,11);
K(2*i,2*r) = K(2*i,2*r) + k(2,12);
```

```
K(2*i,2*s-1) = K(2*i,2*s-1) + k(2,13);
K(2*i,2*s) = K(2*i,2*s) + k(2,14);
K(2*i,2*t-1) = K(2*i,2*t-1) + k(2,15);
K(2*i,2*t) = K(2*i,2*t) + k(2,16);
K(2*j-1,2*i-1) = K(2*j-1,2*i-1) + k(3,1);
K(2*j-1,2*i) = K(2*j-1,2*i) + k(3,2);
K(2*j-1,2*j-1) = K(2*j-1,2*j-1) + k(3,3);
K(2*j-1,2*j) = K(2*j-1,2*j) + k(3,4);
K(2*j-1,2*m-1) = K(2*j-1,2*m-1) + k(3,5);
K(2*j-1,2*m) = K(2*j-1,2*m) + k(3,6);
K(2*j-1,2*p-1) = K(2*j-1,2*p-1) + k(3,7);
K(2*j-1,2*p) = K(2*j-1,2*p) + k(3,8);
K(2*j-1,2*q-1) = K(2*j-1,2*q-1) + k(3,9);
K(2*j-1,2*q) = K(2*j-1,2*q) + k(3,10);
K(2*j-1,2*r-1) = K(2*j-1,2*r-1) + k(3,11);
K(2*j-1,2*r) = K(2*j-1,2*r) + k(3,12);
K(2*j-1,2*s-1) = K(2*j-1,2*s-1) + k(3,13);
K(2*j-1,2*s) = K(2*j-1,2*s) + k(3,14);
K(2*j-1,2*t-1) = K(2*j-1,2*t-1) + k(3,15);
K(2*j-1,2*t) = K(2*j-1,2*t) + k(3,16);
K(2*j,2*i-1) = K(2*j,2*i-1) + k(4,1);
K(2*j,2*i) = K(2*j,2*i) + k(4,2);
K(2*j,2*j-1) = K(2*j,2*j-1) + k(4,3);
K(2*j,2*j) = K(2*j,2*j) + k(4,4);
K(2*j,2*m-1) = K(2*j,2*m-1) + k(4,5);
K(2*j,2*m) = K(2*j,2*m) + k(4,6);
K(2*j,2*p-1) = K(2*j,2*p-1) + k(4,7);
K(2*j,2*p) = K(2*j,2*p) + k(4,8);
K(2*j,2*q-1) = K(2*j,2*q-1) + k(4,9);
K(2*j,2*q) = K(2*j,2*q) + k(4,10);
K(2*j,2*r-1) = K(2*j,2*r-1) + k(4,11);
K(2*j,2*r) = K(2*j,2*r) + k(4,12);
K(2*j,2*s-1) = K(2*j,2*s-1) + k(4,13);
K(2*j,2*s) = K(2*j,2*s) + k(4,14);
K(2*j,2*t-1) = K(2*j,2*t-1) + k(4,15);
K(2*j,2*t) = K(2*j,2*t) + k(4,16);
K(2*m-1,2*i-1) = K(2*m-1,2*i-1) + k(5,1);
K(2*m-1,2*i) = K(2*m-1,2*i) + k(5,2);
K(2*m-1,2*j-1) = K(2*m-1,2*j-1) + k(5,3);
K(2*m-1,2*j) = K(2*m-1,2*j) + k(5,4);
K(2*m-1,2*m-1) = K(2*m-1,2*m-1) + k(5,5);
K(2*m-1,2*m) = K(2*m-1,2*m) + k(5,6);
K(2*m-1,2*p-1) = K(2*m-1,2*p-1) + k(5,7);
K(2*m-1,2*p) = K(2*m-1,2*p) + k(5,8);
K(2*m-1,2*q-1) = K(2*m-1,2*q-1) + k(5,9);
K(2*m-1,2*q) = K(2*m-1,2*q) + k(5,10);
K(2*m-1,2*r-1) = K(2*m-1,2*r-1) + k(5,11);
K(2*m-1,2*r) = K(2*m-1,2*r) + k(5,12);
K(2*m-1,2*s-1) = K(2*m-1,2*s-1) + k(5,13);
K(2*m-1,2*s) = K(2*m-1,2*s) + k(5,14);
K(2*m-1,2*t-1) = K(2*m-1,2*t-1) + k(5,15);
K(2*m-1,2*t) = K(2*m-1,2*t) + k(5,16);
K(2*m,2*i-1) = K(2*m,2*i-1) + k(6,1);
K(2*m,2*i) = K(2*m,2*i) + k(6,2);
K(2*m,2*j-1) = K(2*m,2*j-1) + k(6,3);
K(2*m,2*j) = K(2*m,2*j) + k(6,4);
K(2*m,2*m-1) = K(2*m,2*m-1) + k(6,5);
K(2*m,2*m) = K(2*m,2*m) + k(6,6);
K(2*m,2*p-1) = K(2*m,2*p-1) + k(6,7);
K(2*m,2*p) = K(2*m,2*p) + k(6,8);
K(2*m,2*q-1) = K(2*m,2*q-1) + k(6,9);
K(2*m,2*q) = K(2*m,2*q) + k(6,10);
K(2*m,2*r-1) = K(2*m,2*r-1) + k(6,11);
K(2*m,2*r) = K(2*m,2*r) + k(6,12);
```

```
K(2*m,2*s-1) = K(2*m,2*s-1) + k(6,13);
K(2*m,2*s) = K(2*m,2*s) + k(6,14);
K(2*m,2*t-1) = K(2*m,2*t-1) + k(6,15);
K(2*m,2*t) = K(2*m,2*t) + k(6,16);
K(2*p-1,2*i-1) = K(2*p-1,2*i-1) + k(7,1);
K(2*p-1,2*i) = K(2*p-1,2*i) + k(7,2);
K(2*p-1,2*j-1) = K(2*p-1,2*j-1) + k(7,3);
K(2*p-1,2*j) = K(2*p-1,2*j) + k(7,4);
K(2*p-1,2*m-1) = K(2*p-1,2*m-1) + k(7,5);
K(2*p-1,2*m) = K(2*p-1,2*m) + k(7,6);
K(2*p-1,2*p-1) = K(2*p-1,2*p-1) + k(7,7);
K(2*p-1,2*p) = K(2*p-1,2*p) + k(7,8);
K(2*p-1,2*q-1) = K(2*p-1,2*q-1) + k(7,9);
K(2*p-1,2*q) = K(2*p-1,2*q) + k(7,10);
K(2*p-1,2*r-1) = K(2*p-1,2*r-1) + k(7,11);
K(2*p-1,2*r) = K(2*p-1,2*r) + k(7,12);
K(2*p-1,2*s-1) = K(2*p-1,2*s-1) + k(7,13);
K(2*p-1,2*s) = K(2*p-1,2*s) + k(7,14);
K(2*p-1,2*t-1) = K(2*p-1,2*t-1) + k(7,15);
K(2*p-1,2*t) = K(2*p-1,2*t) + k(7,16);
K(2*p,2*i-1) = K(2*p,2*i-1) + k(8,1);
K(2*p,2*i) = K(2*p,2*i) + k(8,2);
K(2*p,2*j-1) = K(2*p,2*j-1) + k(8,3);
K(2*p,2*j) = K(2*p,2*j) + k(8,4);
K(2*p,2*m-1) = K(2*p,2*m-1) + k(8,5);
K(2*p,2*m) = K(2*p,2*m) + k(8,6);
K(2*p,2*p-1) = K(2*p,2*p-1) + k(8,7);
K(2*p,2*p) = K(2*p,2*p) + k(8,8);
K(2*p,2*q-1) = K(2*p,2*q-1) + k(8,9);
K(2*p,2*q) = K(2*p,2*q) + k(8,10);
K(2*p,2*r-1) = K(2*p,2*r-1) + k(8,11);
K(2*p,2*r) = K(2*p,2*r) + k(8,12);
K(2*p,2*s-1) = K(2*p,2*s-1) + k(8,13);
K(2*p,2*s) = K(2*p,2*s) + k(8,14);
K(2*p,2*t-1) = K(2*p,2*t-1) + k(8,15);
K(2*p,2*t) = K(2*p,2*t) + k(8,16);
K(2*q-1,2*i-1) = K(2*q-1,2*i-1) + k(9,1);
K(2*q-1,2*i) = K(2*q-1,2*i) + k(9,2);
K(2*q-1,2*j-1) = K(2*q-1,2*j-1) + k(9,3);
K(2*q-1,2*j) = K(2*q-1,2*j) + k(9,4);
K(2*q-1,2*m-1) = K(2*q-1,2*m-1) + k(9,5);
K(2*q-1,2*m) = K(2*q-1,2*m) + k(9,6);
K(2*q-1,2*p-1) = K(2*q-1,2*p-1) + k(9,7);
K(2*q-1,2*p) = K(2*q-1,2*p) + k(9,8);
K(2*q-1,2*q-1) = K(2*q-1,2*q-1) + k(9,9);
K(2*q-1,2*q) = K(2*q-1,2*q) + k(9,10);
K(2*q-1,2*r-1) = K(2*q-1,2*r-1) + k(9,11);
K(2*q-1,2*r) = K(2*q-1,2*r) + k(9,12);
K(2*q-1,2*s-1) = K(2*q-1,2*s-1) + k(9,13);
K(2*q-1,2*s) = K(2*q-1,2*s) + k(9,14);
K(2*q-1,2*t-1) = K(2*q-1,2*t-1) + k(9,15);
K(2*q-1,2*t) = K(2*q-1,2*t) + k(9,16);
K(2*q,2*i-1) = K(2*q,2*i-1) + k(10,1);
K(2*q,2*i) = K(2*q,2*i) + k(10,2);
K(2*q,2*j-1) = K(2*q,2*j-1) + k(10,3);
K(2*q,2*j) = K(2*q,2*j) + k(10,4);
K(2*q,2*m-1) = K(2*q,2*m-1) + k(10,5);
K(2*q,2*m) = K(2*q,2*m) + k(10,6);
K(2*q,2*p-1) = K(2*q,2*p-1) + k(10,7);
K(2*q,2*p) = K(2*q,2*p) + k(10,8);
K(2*q,2*q-1) = K(2*q,2*q-1) + k(10,9);
K(2*q,2*q) = K(2*q,2*q) + k(10,10);
K(2*q,2*r-1) = K(2*q,2*r-1) + k(10,11);
K(2*q,2*r) = K(2*q,2*r) + k(10,12);
```

```
K(2*q,2*s-1) = K(2*q,2*s-1) + k(10,13);
K(2*q,2*s) = K(2*q,2*s) + k(10,14);
K(2*q,2*t-1) = K(2*q,2*t-1) + k(10,15);
K(2*q,2*t) = K(2*q,2*t) + k(10,16);
K(2*r-1,2*i-1) = K(2*r-1,2*i-1) + k(11,1);
K(2*r-1,2*i) = K(2*r-1,2*i) + k(11,2);
K(2*r-1,2*j-1) = K(2*r-1,2*j-1) + k(11,3);
K(2*r-1,2*j) = K(2*r-1,2*j) + k(11,4);
K(2*r-1,2*m-1) = K(2*r-1,2*m-1) + k(11,5);
K(2*r-1,2*m) = K(2*r-1,2*m) + k(11,6);
K(2*r-1,2*p-1) = K(2*r-1,2*p-1) + k(11,7);
K(2*r-1,2*p) = K(2*r-1,2*p) + k(11,8);
K(2*r-1,2*q-1) = K(2*r-1,2*q-1) + k(11,9);
K(2*r-1,2*q) = K(2*r-1,2*q) + k(11,10);
K(2*r-1,2*r-1) = K(2*r-1,2*r-1) + k(11,11);
K(2*r-1,2*r) = K(2*r-1,2*r) + k(11,12);
K(2*r-1,2*s-1) = K(2*r-1,2*s-1) + k(11,13);
K(2*r-1,2*s) = K(2*r-1,2*s) + k(11,14);
K(2*r-1,2*t-1) = K(2*r-1,2*t-1) + k(11,15);
K(2*r-1,2*t) = K(2*r-1,2*t) + k(11,16);
K(2*r,2*i-1) = K(2*r,2*i-1) + k(12,1);
K(2*r,2*i) = K(2*r,2*i) + k(12,2);
K(2*r,2*j-1) = K(2*r,2*j-1) + k(12,3);
K(2*r,2*j) = K(2*r,2*j) + k(12,4);
K(2*r,2*m-1) = K(2*r,2*m-1) + k(12,5);
K(2*r,2*m) = K(2*r,2*m) + k(12,6);
K(2*r,2*p-1) = K(2*r,2*p-1) + k(12,7);
K(2*r,2*p) = K(2*r,2*p) + k(12,8);
K(2*r,2*q-1) = K(2*r,2*q-1) + k(12,9);
K(2*r,2*q) = K(2*r,2*q) + k(12,10);
K(2*r,2*r-1) = K(2*r,2*r-1) + k(12,11);
K(2*r,2*r) = K(2*r,2*r) + k(12,12);
K(2*r,2*s-1) = K(2*r,2*s-1) + k(12,13);
K(2*r,2*s) = K(2*r,2*s) + k(12,14);
K(2*r,2*t-1) = K(2*r,2*t-1) + k(12,15);
K(2*r,2*t) = K(2*r,2*t) + k(12,16);
K(2*s-1,2*i-1) = K(2*s-1,2*i-1) + k(13,1);
K(2*s-1,2*i) = K(2*s-1,2*i) + k(13,2);
K(2*s-1,2*j-1) = K(2*s-1,2*j-1) + k(13,3);
K(2*s-1,2*j) = K(2*s-1,2*j) + k(13,4);
K(2*s-1,2*m-1) = K(2*s-1,2*m-1) + k(13,5);
K(2*s-1,2*m) = K(2*s-1,2*m) + k(13,6);
K(2*s-1,2*p-1) = K(2*s-1,2*p-1) + k(13,7);
K(2*s-1,2*p) = K(2*s-1,2*p) + k(13,8);
K(2*s-1,2*q-1) = K(2*s-1,2*q-1) + k(13,9);
K(2*s-1,2*q) = K(2*s-1,2*q) + k(13,10);
K(2*s-1,2*r-1) = K(2*s-1,2*r-1) + k(13,11);
K(2*s-1,2*r) = K(2*s-1,2*r) + k(13,12);
K(2*s-1,2*s-1) = K(2*s-1,2*s-1) + k(13,13);
K(2*s-1,2*s) = K(2*s-1,2*s) + k(13,14);
K(2*s-1,2*t-1) = K(2*s-1,2*t-1) + k(13,15);
K(2*s-1,2*t) = K(2*s-1,2*t) + k(13,16);
K(2*s,2*i-1) = K(2*s,2*i-1) + k(14,1);
K(2*s,2*i) = K(2*s,2*i) + k(14,2);
K(2*s,2*j-1) = K(2*s,2*j-1) + k(14,3);
K(2*s,2*j) = K(2*s,2*j) + k(14,4);
K(2*s,2*m-1) = K(2*s,2*m-1) + k(14,5);
K(2*s,2*m) = K(2*s,2*m) + k(14,6);
K(2*s,2*p-1) = K(2*s,2*p-1) + k(14,7);
K(2*s,2*p) = K(2*s,2*p) + k(14,8);
K(2*s,2*q-1) = K(2*s,2*q-1) + k(14,9);
K(2*s,2*q) = K(2*s,2*q) + k(14,10);
K(2*s,2*r-1) = K(2*s,2*r-1) + k(14,11);
K(2*s,2*r) = K(2*s,2*r) + k(14,12);
```

```
K(2*s,2*s-1) = K(2*s,2*s-1) + k(14,13);
K(2*s,2*s) = K(2*s,2*s) + k(14,14);
K(2*s,2*t-1) = K(2*s,2*t-1) + k(14,15);
K(2*s,2*t) = K(2*s,2*t) + k(14,16);
K(2*t-1,2*i-1) = K(2*t-1,2*i-1) + k(15,1);
K(2*t-1,2*i) = K(2*t-1,2*i) + k(15,2);
K(2*t-1,2*j-1) = K(2*t-1,2*j-1) + k(15,3);
K(2*t-1,2*j) = K(2*t-1,2*j) + k(15,4);
K(2*t-1,2*m-1) = K(2*t-1,2*m-1) + k(15,5);
K(2*t-1,2*m) = K(2*t-1,2*m) + k(15,6);
K(2*t-1,2*p-1) = K(2*t-1,2*p-1) + k(15,7);
K(2*t-1,2*p) = K(2*t-1,2*p) + k(15,8);
K(2*t-1,2*q-1) = K(2*t-1,2*q-1) + k(15,9);
K(2*t-1,2*q) = K(2*t-1,2*q) + k(15,10);
K(2*t-1,2*r-1) = K(2*t-1,2*r-1) + k(15,11);
K(2*t-1,2*r) = K(2*t-1,2*r) + k(15,12);
K(2*t-1,2*s-1) = K(2*t-1,2*s-1) + k(15,13);
K(2*t-1,2*s) = K(2*t-1,2*s) + k(15,14);
K(2*t-1,2*t-1) = K(2*t-1,2*t-1) + k(15,15);
K(2*t-1,2*t) = K(2*t-1,2*t) + k(15,16);
K(2*t,2*i-1) = K(2*t,2*i-1) + k(16,1);
K(2*t,2*i) = K(2*t,2*i) + k(16,2);
K(2*t,2*j-1) = K(2*t,2*j-1) + k(16,3);
K(2*t,2*j) = K(2*t,2*j) + k(16,4);
K(2*t,2*m-1) = K(2*t,2*m-1) + k(16,5);
K(2*t,2*m) = K(2*t,2*m) + k(16,6);
K(2*t,2*p-1) = K(2*t,2*p-1) + k(16,7);
K(2*t,2*p) = K(2*t,2*p) + k(16,8);
K(2*t,2*q-1) = K(2*t,2*q-1) + k(16,9);
K(2*t,2*q) = K(2*t,2*q) + k(16,10);
K(2*t,2*r-1) = K(2*t,2*r-1) + k(16,11);
K(2*t,2*r) = K(2*t,2*r) + k(16,12);
K(2*t,2*s-1) = K(2*t,2*s-1) + k(16,13);
K(2*t,2*s) = K(2*t,2*s) + k(16,14);
K(2*t,2*t-1) = K(2*t,2*t-1) + k(16,15);
K(2*t,2*t) = K(2*t,2*t) + k(16,16);
y = K;
end
```

#### 7.4.7.5 *Stress Calculation Q8*

```
function [ sigma,sigmaX, sigmaBX, xB, yB, Qtop ] =
StressCalculationQ8( E,NU,U,Lx,Ly,dx,dy,element,p,xe,ye )
%StressDistribution     This function returns a graph between the
stress values of the beam and its length. E is the modulus of
% elasticity, NU is Poisson's ratio, U is the displacement vector of
%the beam, Lx is the Length of the beam, Ly is the beam's height, dx
%is %the length of the mesh in x-direction, dy is the length of the
%mesh in y-direction, element is a matrix contains information about
%the nodes and their coordinates, p is the problem that is going to be
%uses (1 for stress and 2 for strain) and finally xe and ye are the
%coordinates of the point that the stresses are going to be calculates
nx = Lx/dx;
ny = Ly/dy;
no = 0;
for q = 1: size(element,1)
    a = element(q,2);
    b = element(q,3);
    c = element(q,4);
    d = element(q,5);
    ab = element(q,6);
    bc = element(q,7);
    cd = element(q,8);
    da = element(q,9);
    x1 = element(q,10);
    y1 = element(q,11);
    x2 = element(q,12);
    y2 = element(q,13);
    x3 = element(q,14);
    y3 = element(q,15);
    x4 = element(q,16);
    y4 = element(q,17);
    uu((16*no)+1:(16*no)+16,1)=[U((((a*2)-1)):(((a*2)+2)));
    U(((c*2)-1));U((c*2));U(((d*2)-1));U((d*2));U((ab*2)-
1,1);U(ab*2,1);U((bc*2)-1,1);U(bc*2,1);U((cd*2)-
51,1);U(cd*2,1);U((da*2)-1,1);U(da*2,1)];
%sigma is the stresses at xe = [-dx/2 and dx/2] & ye = 0 to draw a
%distribution of the normal stresses along the beam and compare it to
%the exact ones
    sigma((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),xe,ye);
    %to calculate shear stress
    sigmaQcentroid((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),xe,0);
    %sigma0 is the stresses at the centroid of the element
    sigma0((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),0,0);
    %sigmabottom is the stresses of all elements at their bottom side
    sigmabottom((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),-dx/2,-dy/2);
    %sigmatop is the stesses of all elements at their top right side
    sigmatop((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),dx/2,dy/2);
    %sigmaleft is the stresses of all elements at their top left side
    sigmaleft((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),-dx/2,dy/2);
    %sigma right is the stresses of all elements at their bottom right
side
    sigmaright((3*no)+1:(3*no)+3,:) =
QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,uu((16*no)
+1:(16*no)+16,1),dx/2,-dy/2);
    no = no+1;
end

%n are the number of elements used
n = size(element,1);

%sigmaBX are the normal stresses at the boundary nodes of the beams
sigma0X = sigma0(1:3:size(sigma0),:);
sigmatopX = sigmatop(1:3:size(sigmatop),:);
sigmabottomX = sigmabottom(1:3:size(sigmabottom),:);
sigmaleftX = sigmaleft(1:3:size(sigmaleft),:);
sigmarightX = sigmaright(1:3:size(sigmaright),:);

x1 = element(:,10);
y1 = element(:,11);
x2 = element(:,12);
```

```
y2 = element(:,13);
x3 = element(:,14);
y3 = element(:,15);
x4 = element(:,16);
y4 = element(:,17);
sigmaBbottomX = sigmabottomX(1:nx,:);
xbottom = x1(1:nx,:);
ybottom = y1(1:nx,:);
sigmaBtopX = sigmatopX(n-nx+1:n,:);
xtop = x3(n-nx+1:n,:);
ytop = y3(n-nx+1:n,:);
sigmaBleftX = sigmaleftX(1:nx:n,:);
xleft = x4(1:nx:n,:);
yleft = y4(1:nx:n,:);
sigmaBrightX = sigmarightX(nx:nx:n,:);
xright = x2(nx:nx:n,:);
yright = y2(nx:nx:n,:);

sigmaBX = [sigma0X; sigmaBbottomX; sigmaBtopX; sigmaBleftX;
sigmaBrightX];
xB = [xbottom; xtop; xleft; xright];
yB = [ybottom; ytop; yleft; yright];

%sigmaX are the normal stresses
sigmaX = sigma(1:3:size(sigma),:);

%Q is the shear stress
Qcentroid = sigmaQcentroid(3:3:size(sigmaQcentroid),:);
Qtop = sigma(3:3:size(sigma),:);
end
```

### *7.4.7.6 Quadratic Quadrature Element Stresses*

```
function w
=QuadraticQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,u,xe,ye)
%QuadraticQuadElementStresses This function returns the element
% stress vector for a quadratic quadrilateral element with modulus
% of elasticity E, Poisson's ratio NU, coordinates of
% node 1 (x1,y1), coordinates of node 2 (x2,y2), coordinates of
% node 3 (x3,y3), coordinates of node 4 (x4,y4), and element
% displacement vector u. Use p = 1 for cases
% of plane stress, and p = 2 for cases of plane strain.
syms s t;
x5 = (x1 + x2)/2;
x6 = (x2 + x3)/2;
x7 = (x3 + x4)/2;
x8 = (x4 + x1)/2;
y5 = (y1 + y2)/2;
y6 = (y2 + y3)/2;
y7 = (y3 + y4)/2;
y8 = (y4 + y1)/2;
N1 = (1-s)*(1-t)*(-s-t-1)/4;
N2 = (1+s)*(1-t)*(s-t-1)/4;
N3 = (1+s)*(1+t)*(s+t-1)/4;
N4 = (1-s)*(1+t)*(-s+t-1)/4;
N5 = (1-t)*(1+s)*(1-s)/2;
N6 = (1+s)*(1+t)*(1-t)/2;
N7 = (1+t)*(1+s)*(1-s)/2;
N8 = (1-s)*(1+t)*(1-t)/2;
x = N1*x1 + N2*x2 + N3*x3 + N4*x4 + N5*x5 + N6*x6 + N7*x7 + N8*x8;
y = N1*y1 + N2*y2 + N3*y3 + N4*y4 + N5*y5 + N6*y6 + N7*y7 + N8*y8;
```

```
xs = diff(x,s);
xt = diff(x,t);
ys = diff(y,s);
yt = diff(y,t);
J = xs*yt - ys*xt;
N1s = diff(N1,s);
N2s = diff(N2,s);
N3s = diff(N3,s);
N4s = diff(N4,s);
N5s = diff(N5,s);
N6s = diff(N6,s);
N7s = diff(N7,s);
N8s = diff(N8,s);
N1t = diff(N1,t);
N2t = diff(N2,t);
N3t = diff(N3,t);
N4t = diff(N4,t);
N5t = diff(N5,t);
N6t = diff(N6,t);
N7t = diff(N7,t);
N8t = diff(N8,t);
B11 = yt*N1s - ys*N1t;
B12 = 0;
B13 = yt*N2s - ys*N2t;
B14 = 0;
B15 = yt*N3s - ys*N3t;
B16 = 0;
B17 = yt*N4s - ys*N4t;
B18 = 0;
B19 = yt*N5s - ys*N5t;
B110 = 0;
B111 = yt*N6s - ys*N6t;
B112 = 0;
B113 = yt*N7s - ys*N7t;
B114 = 0;
B115 = yt*N8s - ys*N8t;
B116 = 0;
B21 = 0;
B22 = xs*N1t - xt*N1s;
B23 = 0;
B24 = xs*N2t - xt*N2s;
B25 = 0;
B26 = xs*N3t - xt*N3s;
B27 = 0;
B28 = xs*N4t - xt*N4s;
B29 = 0;
B210 = xs*N5t - xt*N5s;
B211 = 0;
B212 = xs*N6t - xt*N6s;
B213 = 0;
B214 = xs*N7t - xt*N7s;
B215 = 0;
B216 = xs*N8t - xt*N8s;
B31 = xs*N1t - xt*N1s;
B32 = yt*N1s - ys*N1t;
B33 = xs*N2t - xt*N2s;
B34 = yt*N2s - ys*N2t;
B35 = xs*N3t - xt*N3s;
B36 = yt*N3s - ys*N3t;
B37 = xs*N4t - xt*N4s;
B38 = yt*N4s - ys*N4t;
B39 = xs*N5t - xt*N5s;
B310 = yt*N5s - ys*N5t;
B311 = xs*N6t - xt*N6s;
```

```
B312 = yt*N6s - ys*N6t;
B313 = xs*N7t - xt*N7s;
B314 = yt*N7s - ys*N7t;
B315 = xs*N8t - xt*N8s;
B316 = yt*N8s - ys*N8t;
Jnew = simplify(J);
B = [B11 B12 B13 B14 B15 B16 B17 B18 B19 B110 B111 B112 B113 B114 B115
B116;B21 B22 B23 B24 B25 B26 B27 B28 B29 B210 B211 B212 B213 B214 B215
B216; B31 B32 B33 B34 B35 B36 B37 B38 B39 B310 B311 B312 B313 B314
B315 B316]/Jnew;
if p == 1
D = (E/(1-NU*NU))*[1, NU, 0 ; NU, 1, 0 ; 0, 0, (1-NU)/2];
elseif p == 2
D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0 ;NU, 1-NU, 0 ; 0, 0, (1-2*NU)/2];
end
Bnew = simplify(B);
w = D*Bnew*u;
% We calculate the stresses at (xe,ye) of the element
wcent = subs(w, {s,t}, {xe,ye});
w = double(wcent);
end
```

#### *7.4.7.7 Draw Beam Q8*

```
function y = DrawBeamQ8( element )
%DrawBeam This function returns a graph for an undefromed beam after
%meshing with 8-node elements. The input is element matrix

hold on
for ii = 1: size(element,1)
    x1 = element(ii,10);
    y1 = element(ii,11);
    x2 = element(ii,12);
    y2 = element(ii,13);
    x3 = element(ii,14);
    y3 = element(ii,15);
    x4 = element(ii,16);
    y4 = element(ii,17);
    plot([x1, x2], [y1, y2],'color',[1 0 0]);
    plot([x2, x3], [y2, y3],'color',[1 0 0]);
    plot([x3, x4], [y3, y4],'color',[1 0 0]);
    plot([x4, x1], [y4, y1],'color',[1 0 0]);
end
hold off
end
```

#### *7.4.7.8 Draw Deformed Beam Q8*

```
function [ ] = DrawDeformedBeamQ8( element,U )
%DrawDeformedBeam This function returns a graph for a beam modeled by
%Q8 element after deformation. The input is element matrix and
%Displacement vector U. (It is better to multiply U vector by certain
%value for better graph.

hold on
axis equal
for ii = 1: size(element,1)
    i = element(ii,2);
    j = element(ii,3);
```

```
m = element(ii,4);
n = element(ii,5);
ij = element(ii,6);
jm = element(ii,7);
mn = element(ii,8);
ni = element(ii,9);
x1 = element(ii,10);
y1 = element(ii,11);
x2 = element(ii,12);
y2 = element(ii,13);
x3 = element(ii,14);
y3 = element(ii,15);
x4 = element(ii,16);
y4 = element(ii,17);
x5 = (x1 + x2)/2;
x6 = (x2 + x3)/2;
x7 = (x3 + x4)/2;
x8 = (x4 + x1)/2;
y5 = (y1 + y2)/2;
y6 = (y2 + y3)/2;
y7 = (y3 + y4)/2;
y8 = (y4 + y1)/2;
ux1 = U((i*2)-1,1);
ux2 = U((j*2)-1,1);
ux3 = U((m*2)-1,1);
ux4 = U((n*2)-1,1);
ux5 = U((ij*2)-1,1);
ux6 = U((jm*2)-1,1);
ux7 = U((mn*2)-1,1);
ux8 = U((ni*2)-1,1);
uy1 = U((i*2),1);
uy2 = U((j*2),1);
uy3 = U((m*2),1);
uy4 = U((n*2),1);
uy5 = U((ij*2),1);
uy6 = U((jm*2),1);
uy7 = U((mn*2),1);
uy8 = U((ni*2),1);
x1new = x1+ux1;
x2new = x2+ux2;
x3new = x3+ux3;
x4new = x4+ux4;
x5new = x5+ux5;
x6new = x6+ux6;
x7new = x7+ux7;
x8new = x8+ux8;
y1new = y1+uy1;
y2new = y2+uy2;
y3new = y3+uy3;
y4new = y4+uy4;
y5new = y5+uy5;
y6new = y6+uy6;
y7new = y7+uy7;
y8new = y8+uy8;
centroidnew(ii,:) = [(((x2new+x1new)/2)+((x3new)...
    +x4new)/2)/2,(((y3new+y2new)/2)+((y1new...
    +y4new))/2)/2];
plot([x1new, x5new], [y1new, y5new],':','color',[0 1 0]);
plot([x5new, x2new], [y5new, y2new],':','color',[0 1 0]);
plot([x2new, x6new], [y2new, y6new],':','color',[0 1 0]);
plot([x6new, x3new], [y6new, y3new],':','color',[0 1 0]);
plot([x3new, x7new], [y3new, y7new],':','color',[0 1 0]);
plot([x7new, x4new], [y7new, y4new],':','color',[0 1 0]);
plot([x4new, x8new], [y4new, y8new],':','color',[0 1 0]);
```

```
    plot([x8new, x1new], [y8new, y1new],':','color',[0 1 0]);
end
axis tight
xlabel('Length of beam (m)')
ylabel('Depth of beam (m)')
title('Deformed shape of the beam
Q8','FontSize',12,'FontWeight','Bold')
hold off
end
```

## 7.5 Appendix E: Q6 element

### 7.5.1 *CantileverBeam_C2_01_03_Q6*

```
%.....................................................................
%Cantilever_C2_01_03_Q6
%This script calculates the normal and shear stresses of a cantilever
beam by 1
%concentrated load at its edge
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modolus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
numbering of the
%nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
f(28,1) = -25;
f(14,1) = -25;
f(15,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;  %k is the stiffness matrix
k(15,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,15) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([15;2;1],u);  %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (?x, ?y and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses ?x at the boundary nodes of the beam
%xB & yB are the positions of the sigmaBX stresses
%Qc is the shear stresses at the centroid of the beam
%Q0 is the min shear stresses which is at the top elements of the beam
```

```
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculationQ6(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);

%Draw the normal stress
figure(2)
DrawStresses(Lx, dx, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeamQ6(element)
DrawDeformedBeamQ6(element,U*1000);
hold off

%Exact analysis
CantileverBeam_C1
%.....................................................................
```

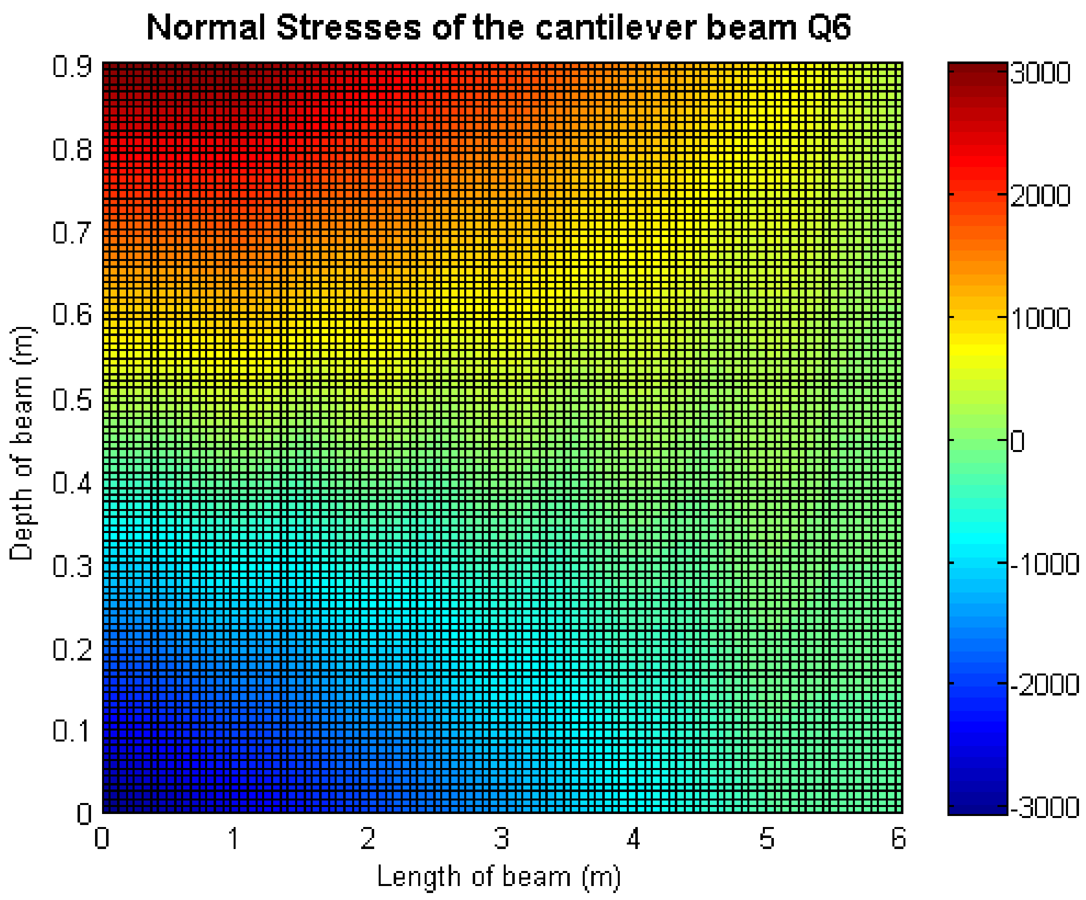


**Fig. 7-13** CantileverBeam_C2_01_03_Q6

### *7.5.2 CantileverBeam_C2_02_03_Q6*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.5;   %Length of the element in x direction
dy = 0.225; %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
f(65*2,1) = -50/(ny+1);
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 53;
for i = 1: ny+1
f(no*2,:) = [];
f((no*2)-1,:) = [];
k(no*2,:) = [];
k((no*2)-1,:) = [];
k(:,no*2) = [];
k(:,(no*2)-1)=[];
noo((i*2),1) = no*2;
noo((i*2)-1,1) = (no*2)-1;
no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement(noo,u);
F = K*U;
```

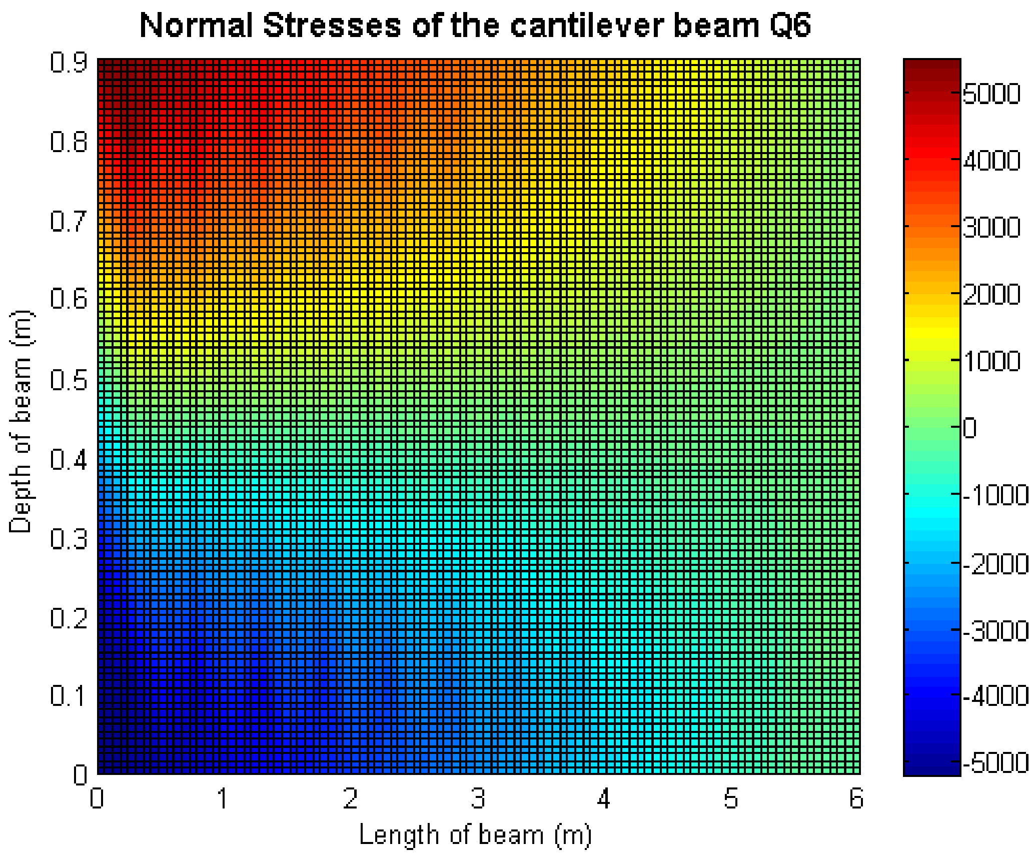


**Fig. 7-14** CantileverBeam_C2_02_03_Q6

### *7.5.3 CantileverBeam_C2_03_03_Q6*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

```
%Meshing
dx = 0.1;   %Length of the element in x direction
dy = 0.05;  %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
f(1159*2,1) = -50/(ny+1);
iino = nx+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;
no = 1099;
for i = 1: ny+1
f(no*2,:) = [];
```

```
f((no*2)-1,:) = [];
k(no*2,:) = [];
k((no*2)-1,:) = [];
k(:,no*2) = [];
k(:,(no*2)-1)=[];
noo((i*2),1) = no*2;
noo((i*2)-1,1) = (no*2)-1;
no = no - nx - 1;
end
u = k\f;
U = InsertZeroElement(noo,u);
F = K*U;
```

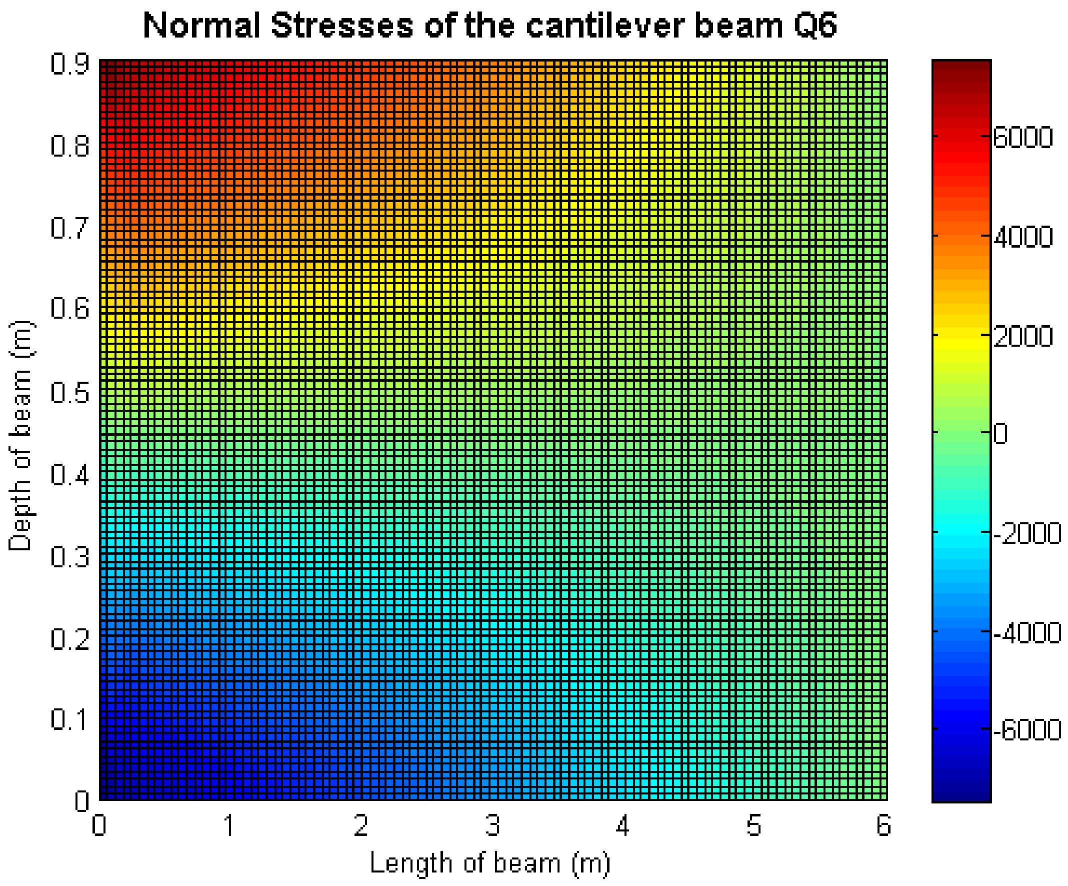
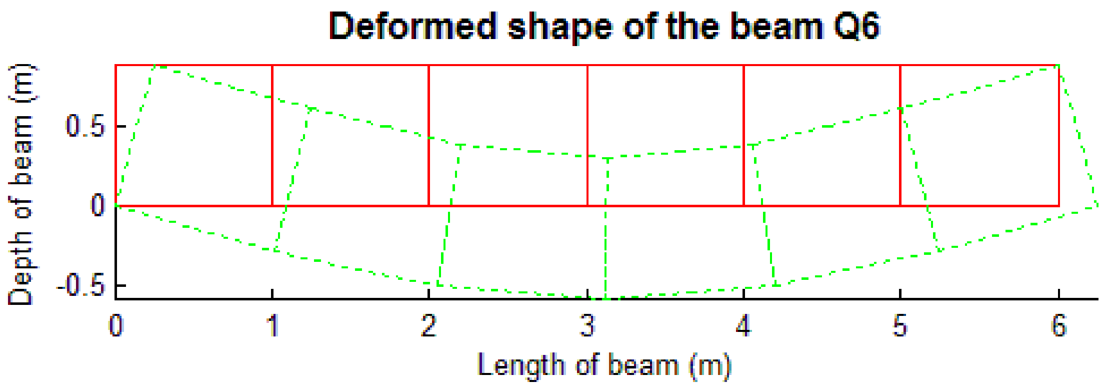


**Fig. 7-15** CantileverBeam_C2_03_03_Q6

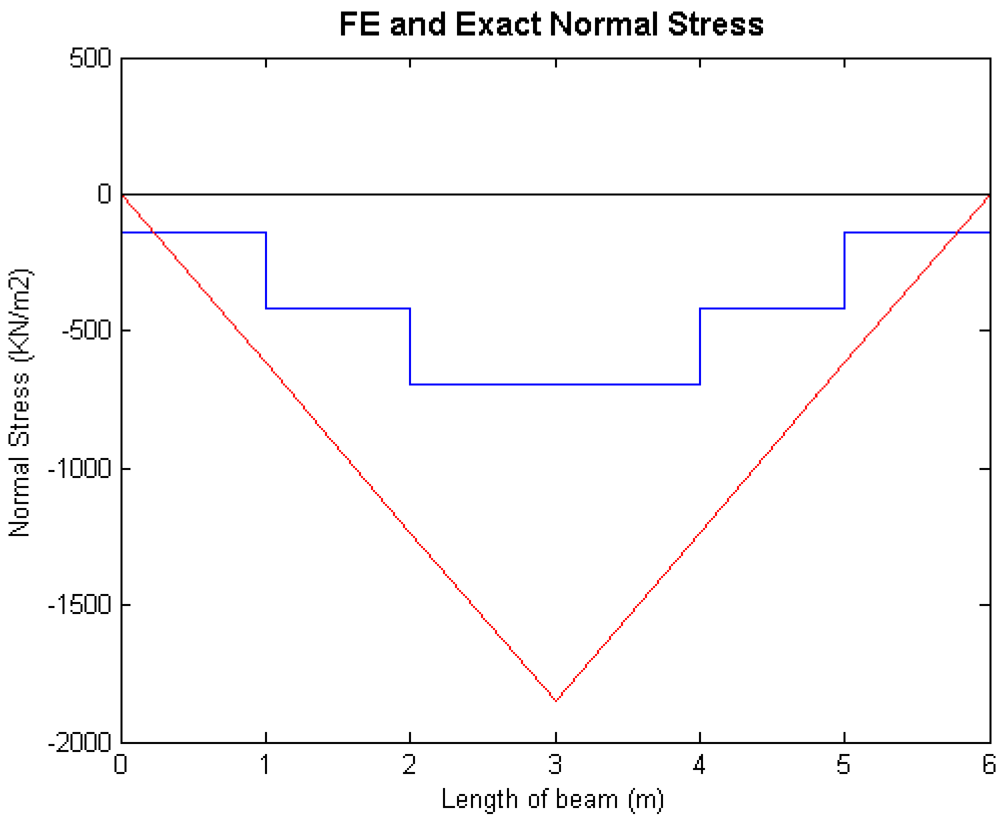
### *7.5.4 SimpleBeam_A2_01_03_Q6*

```
%.....................................................................
%SimpleBeam_A2_01_03_Q6
%This script calculates the normal and shear stresses of a simple beam
%by 1 concentrated load at mid-span
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
```

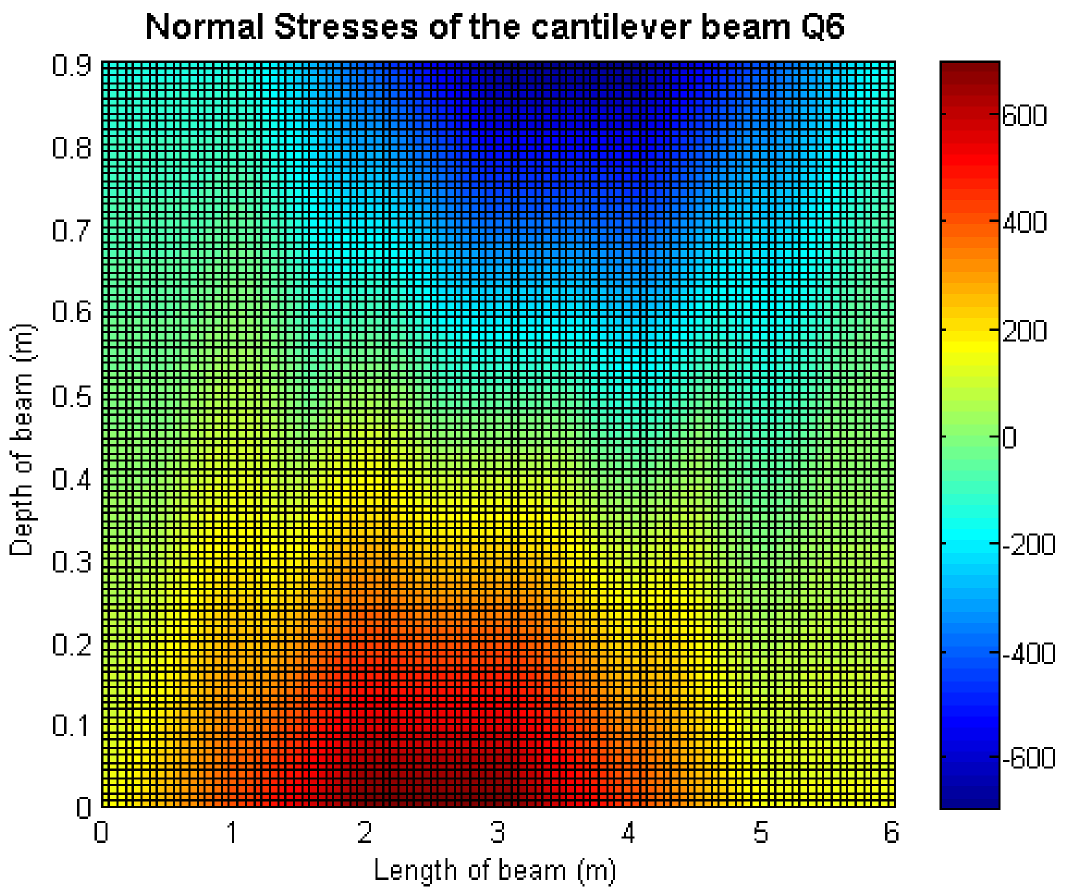
```
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 1; %Length of the element in x direction
dy = 0.9;   %Length of the element in y direction

%Global Stiffness Matrix K
%element is a matrix contains all the data of each element (ID,
numbering of the
%nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
f(22,1) = -25;
f(8,1) = -25;
f(14,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;  %k is the stiffness matrix
k(14,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,14) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([14;2;1],u);  %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (?x, ?y and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses ?x at the boundary nodes of the beam
%xB & yB are the positions of the sigmaBX stresses
%Qc is the shear stresses at the centroid of the beam
%Q0 is the min shear stresses which is at the top elements of the beam
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculationQ6(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);

%Draw the normal stress
figure(2)
DrawStresses(Lx, dx, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeamQ6(element)
DrawDeformedBeamQ6(element,U*10000);
hold off

%Exact analysis
SimpleBeam_A1
%.....................................................................
```


Deformed shape of the beam Q6
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

(a)


FE and Exact Normal Stress
500
0
-500
-1000
-1500
-2000
Normal Stress (KN/m2)
0
1
2
3
4
5
6
Length of beam (m)

(b)



(c)

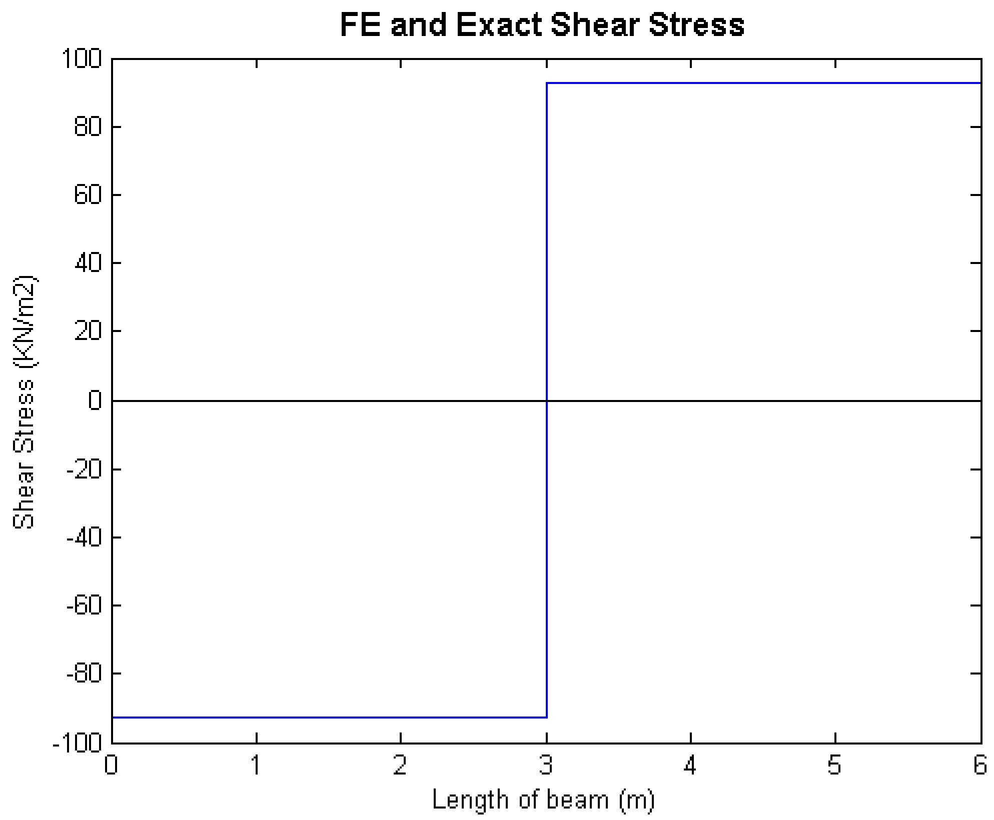
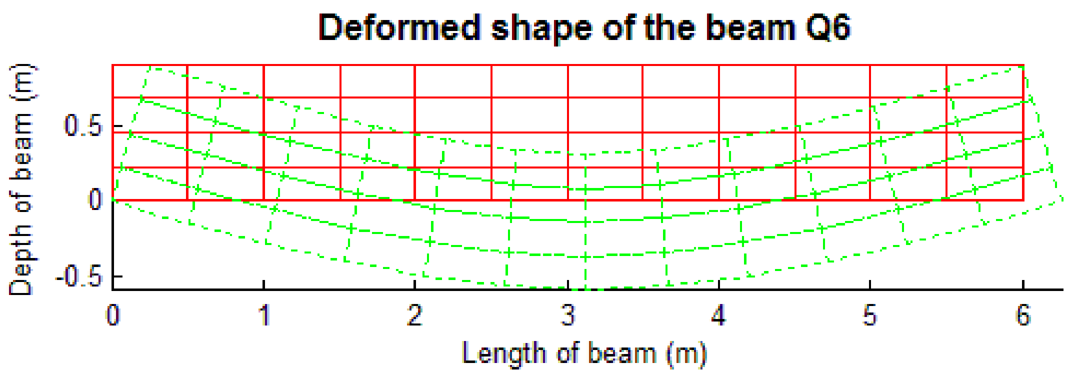


**Fig. 7-16** Analysis of SimpleBeam_A2_01_03_Q6

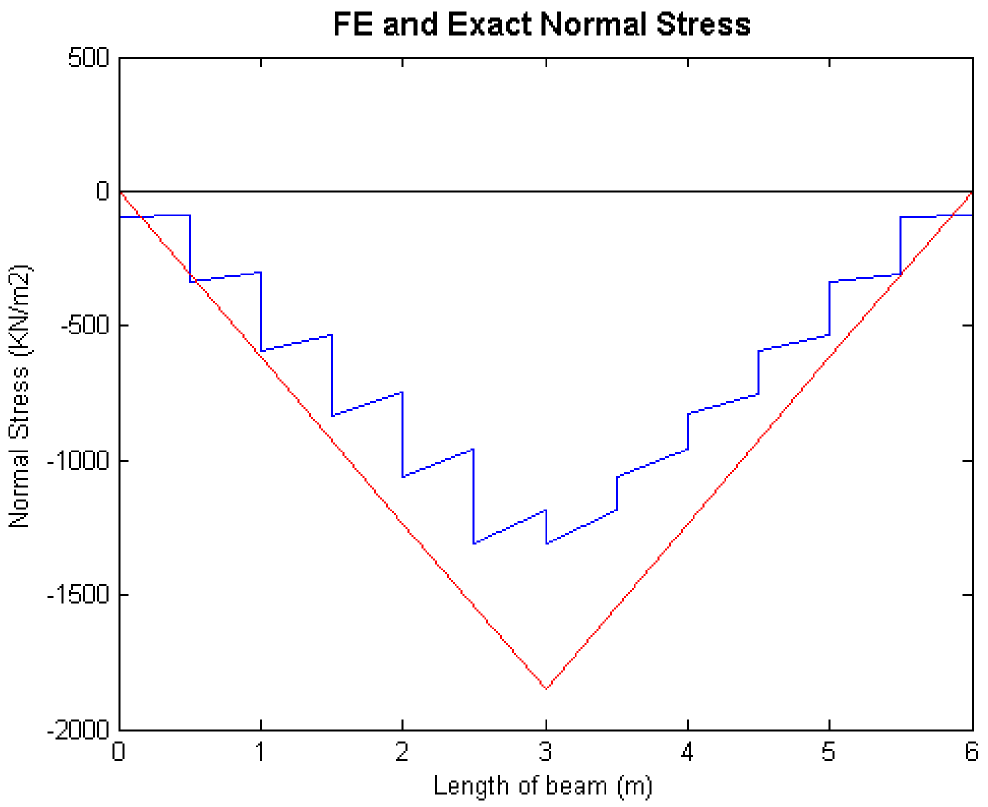
### 7.5.5 *SimpleBeam_A2_02_03_Q6*

```
%......................................................................
%SimpleBeam_A2_02_03_Q6
%This script calculates the normal and shear stresses of a simple beam
%by 1 concentrated load at mid-span
%All units are in metric KN m

%Clear memory
clear all;
clc;
close all;

%Properties of the beam's material and cross section
E = 210e6;  %Modulus of Elasticity E
NU = 0.3;   %Poisson's ratio NU
%Dimension of the beam
Lx = 6; %the length of beam
Ly = 0.9;   %depth
t = 0.3;    %thickness

%Meshing
dx = 0.5;   %Length of the element in x direction
dy = 0.225; %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
nx = Lx/dx;
ny = Ly/dy;
%Dividing the nodal forces along the elements on the left side
f(59*2,1) = -50/(ny+1);
iino = (nx/2)+1;
for ii = 1:ny+1;
    f(iino*2,1) = -50/(ny+1);
    iino = iino+nx+1;
end
k = K;  %k is the stiffness matrix
no = 53;
%Removing all the unknown forces and the corresponding rows and
%columns in the global stiffness matrix
f(26,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(26,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,26) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;26],u);   %U is the displacement vector
F = K*U;    %F is the force vector

%Calculation of the stresses sigma
%sigma is the stresses (sx, sy and shear stress).
%sigmaX is the normal stresses in x-direction only.
%sigmaBX is the normal stresses ?x at the boundary nodes of the beam
```

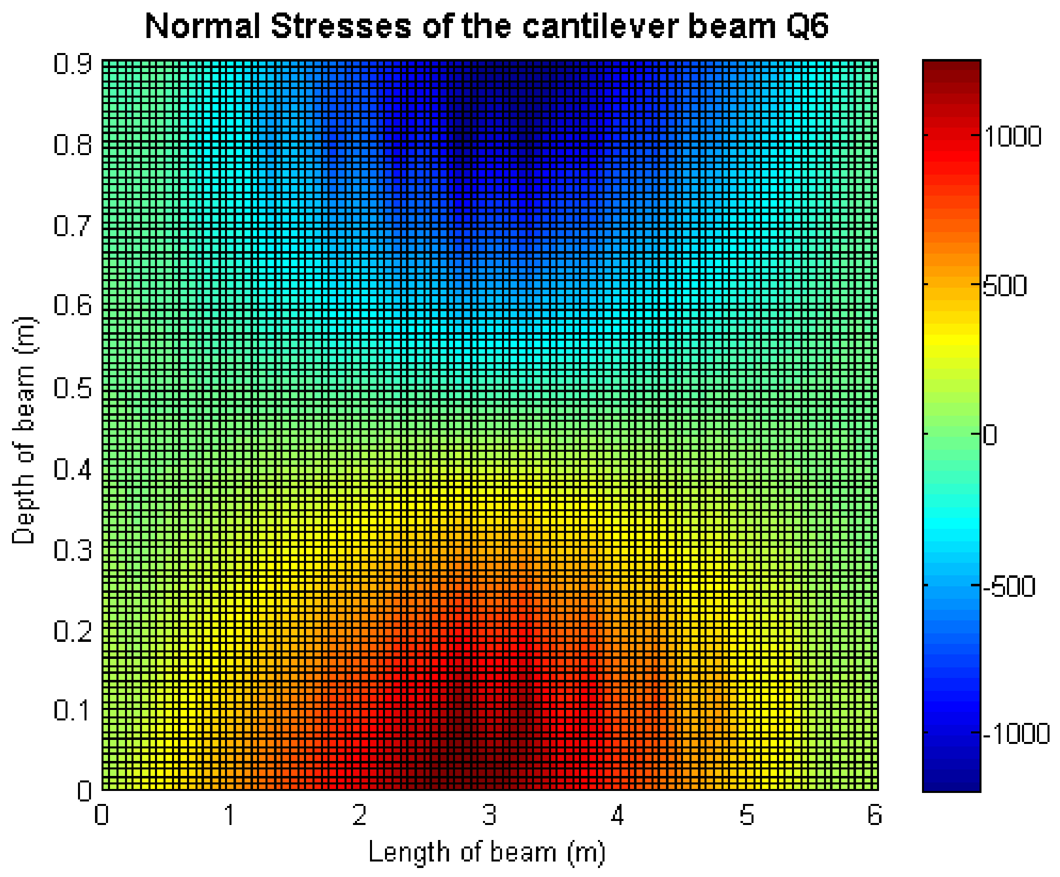
```
%xB & yB are the positions of the sigmaBX stresses
%Q0 is the min shear stresses which is at the top elements of the beam
[sigma, sigmaX, sigmaBX, xB, yB, Qc, Q0] =
StressCalculationQ6(E,NU,U,Lx,Ly,dx,dy,element,1,[-dx/2,dx/2],dy/2);
%Draw the normal stress
figure(2)
DrawStresses(Lx, dx, element, sigmaX);
%Draw the shear stress
figure(4)
DrawStresses(Lx, dx, element, Qc);
%Contours for normal stress
figure(5)
DrawContoursNormalStresses(sigmaBX, centroid, xB, yB, Lx, Ly, 100)
%Deformed Shape of the beam
figure(6)
hold on
DrawBeamQ6(element)
DrawDeformedBeamQ6(element,U*10000);
hold off
%Exact analysis
SimpleBeam_A1
%.....................................................................
```



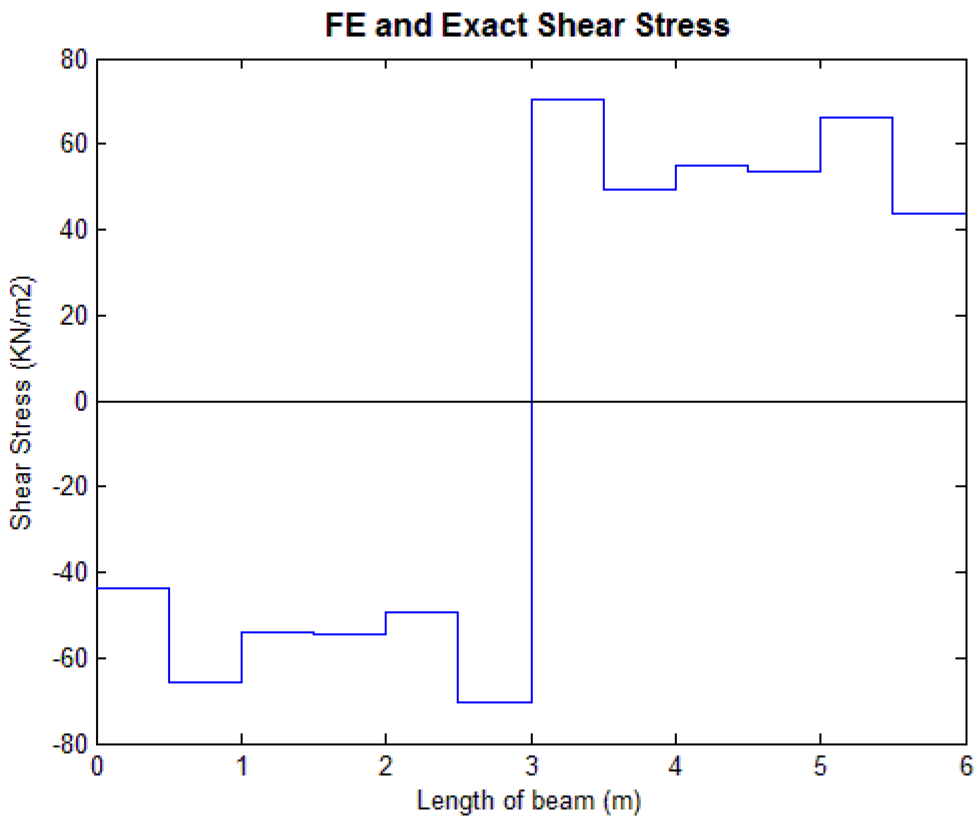
(a)



(b)



(c)



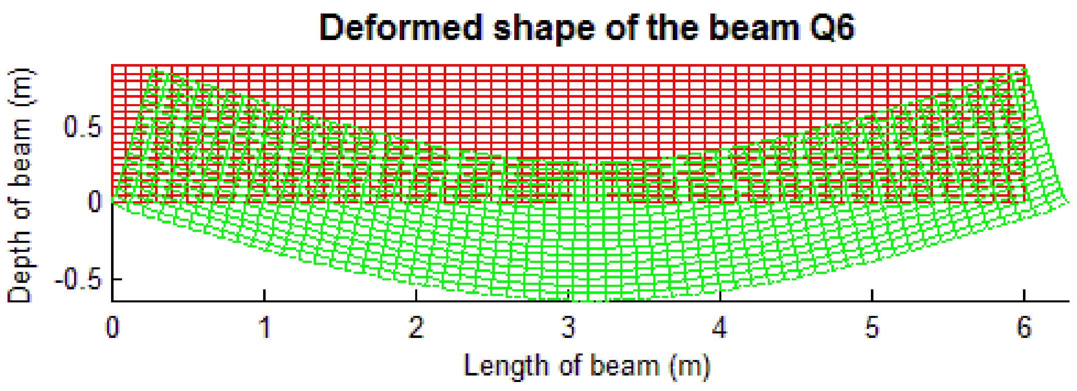
(d)

**Fig. 7-17** Analysis of SimpleBeam_A2_02_03_Q6

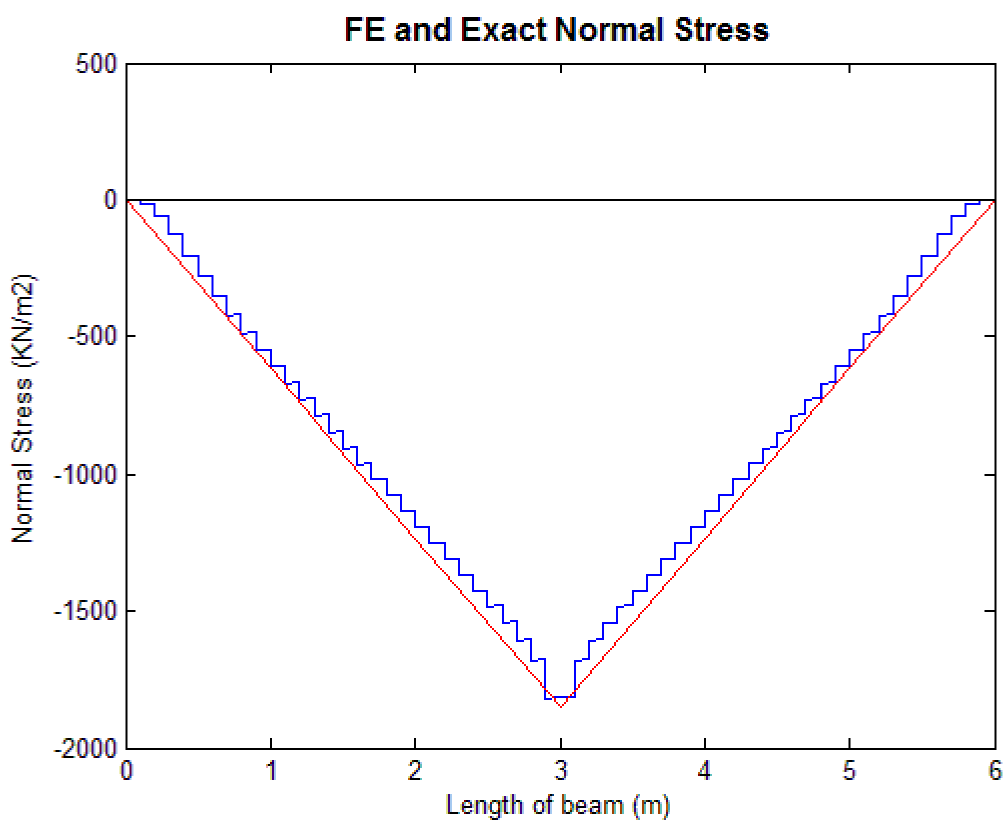
### *7.5.6 SimpleBeam_A2_03_03_Q6*

The previous script is repeated. Only the meshing and Global Stiffness Matrix parts are changed to the following.

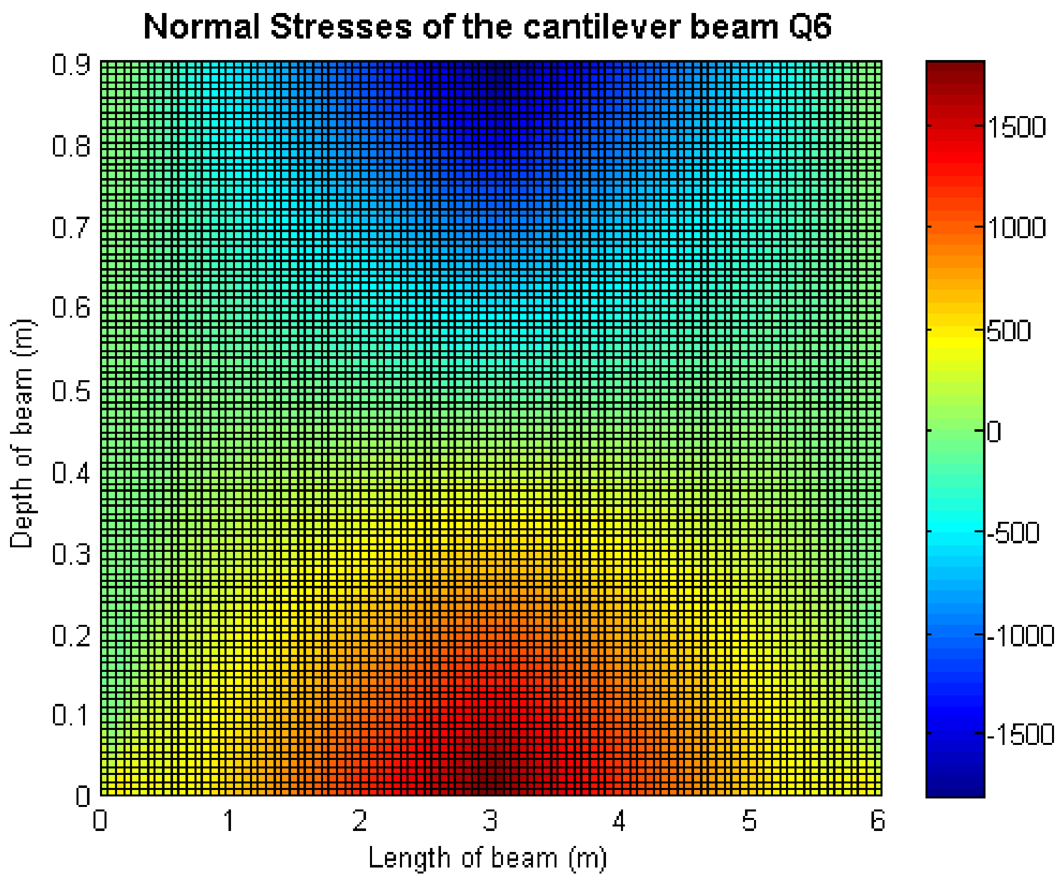
```
%Meshing
dx = 0.1;   %Length of the element in x direction
dy = 0.05;  %Length of the element in y direction

%Global Stiffness Matrix K
%Matrix element contains all the data of each element (ID, numbering
%of the nodes and coordinates)
[element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1);
f = zeros(size(K),1);
ny = Ly/dy;
f(31*2,1) = -50/(ny+1);
iino = 31*2;
for ii = 1:ny;
    iino = iino+(61*2);
    f(iino) = -50/(ny+1);
end
f(122,:) = [];
f(2,:) = [];
f(1,:) = [];
k = K;
k(122,:) = [];
k(2,:) = [];
k(1,:) = [];
k(:,122) = [];
k(:,2) = [];
k(:,1)=[];
u = k\f;
U = InsertZeroElement([1;2;122],u);
F = K*U;
```



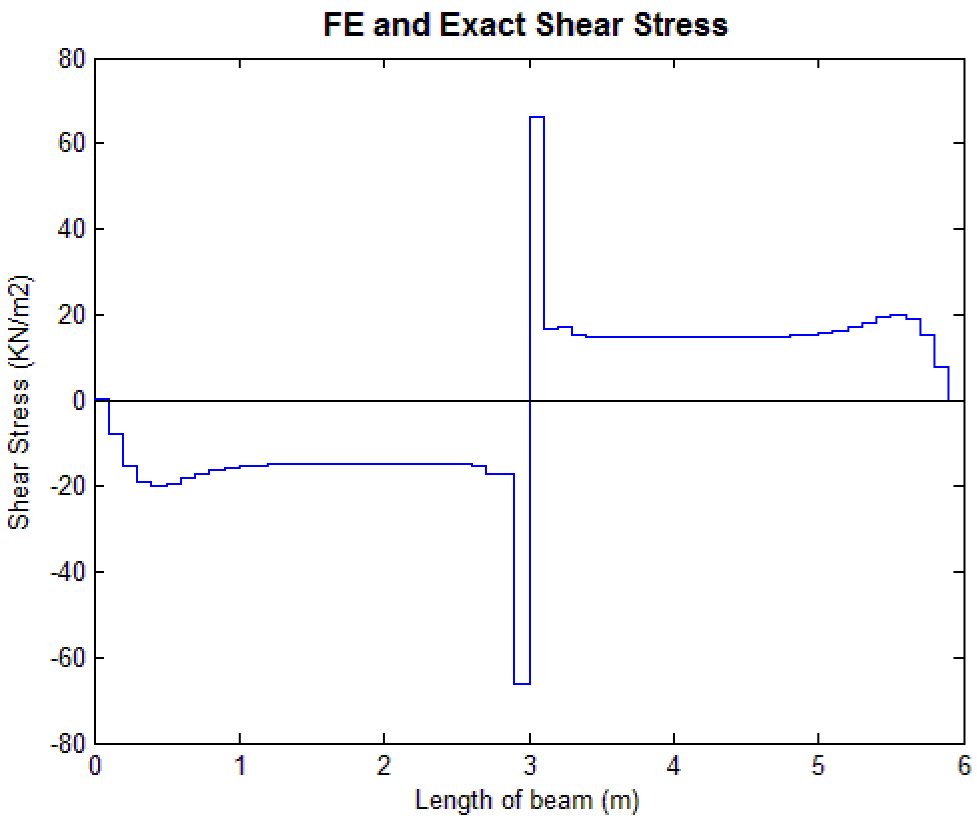
(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
Length of beam (m)
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6

(b)


Normal Stresses of the cantilever beam Q6
Depth of beam (m)
Length of beam (m)
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
2
3
4
5
6
1500
1000
500
0
-500
-1000
-1500

(c)



(d)

**Fig. 7-18** Analysis of SimpleBeam_A2_03_03_Q6

### *7.5.7 Codes used in the script*

#### *7.5.7.1 Improved Bilinear Quadratic Element Stiffness Assembly*

```
function [element,K,centroid] =
ImprovedBilinearQuadElementStiffnessAssembly(E,NU,h,Lx,Ly,dx,dy,p)
%ImprovedBilinearQuadElementStiffnessAssembly This code returns the
%Global stiffness matrix for a beam/plate to be analyzed as
%incompitable element.
%   The inputs are modulus of elasticity E, Poisson's ratio NU,
%thickness of the element t, length of the beam Lx, its depth Ly,
%length of element in x dx, length of the element in y Ly and p which
%is 1 in case of plain stress problems and 2 in case of plain strain
%problems. The output is the element matrix which contains all the
%details of each element for the beam, the whole global stiffness
%matrix for the whole beam K and the a matrix which contains the
%positions of the centroid of the elements centroid.
[element,nx,ny] = BeamMeshingQ6(Lx,Ly,dx,dy);

no = 0;
K =
zeros((2*(nx+1)*(ny+1))+(nx*(ny+1))+(ny*(nx+1)),2*(nx+1)*(ny+1)+(nx*(n
y+1))+(ny*(nx+1)));
for ii=1:size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    ij = element(ii,6);
    jm = element(ii,7);
    mn = element(ii,8);
    ni = element(ii,9);
    x1 = element(ii,10);
    y1 = element(ii,11);
    x2 = element(ii,12);
    y2 = element(ii,13);
    x3 = element(ii,14);
    y3 = element(ii,15);
    x4 = element(ii,16);
    y4 = element(ii,17);
    centroid(ii,:) = [(x2+x1)/2,(y3+y2)/2];
    k =
ImprovedIIBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,y4,
p);
    no = no + 1;
    K = ImprovedBilinearQuadAssemble(K,k,i,j,m,n,ij,jm,mn,ni);
end
```

### *7.5.7.2 BeamMeshingQ6*

```
function [ element,nx,ny ] = BeamMeshingQ6( Lx,Ly,dx,dy )
%BeamMeshing    This function returns element matrix which contains ID
%of elements, ID of nodes of each element and the coordinates of each
%node for each element. element matrix is (number of elements*17).
%  The inputs are the length of the beam Lx, its depth Ly, length of
%the element in x direction dx and its length in y dy. This code is
%done by certain algorithm to have an automatic organized meshing for
%a 4-noded element.
nx = Lx/dx; %number of elements in x direction
ny = Ly/dy; %number of elements in y direction
elements = nx*ny;   %total number of elements.

for r = 1:elements+ny
    a(r,1:4) = [r r+1 nx+2+r nx+1+r];
end
rno = (elements+nx+ny+1)*2;
for r = 1:elements+nx*(ny-1)+(ny-1)
    d(r,1:4) = [rno+r rno+r+nx+1 rno+r+2*nx+1 rno+r+nx];
end
a(nx+1:nx+1:size(a,1),:) = [];  %matrix a has to do with the indices
                                %of the node


for rr = nx+1: nx: size(a,1)
    for rrr = nx+1: 2*nx+1
        d(rr,:) = [];
    end
end

n = 0;
m = 1;
for c = 1:ny
    nn = 0;
    for cc = 1:nx+1
        b(m,1:8) = [nn*dx n*dy cc*dx n*dy cc*dx c*dy nn*dx c*dy];
        nn = 1+nn;
        m = m+1;
    end
    n = n+1;
end

b(nx+1:nx+1:size(b,1),:) = [];  %matrix b has to do with the
                                %corresponding positions for each node
for id = 1:elements
    idd(id,1) = id; %the index of the element
end
element=[idd a d b];    %element matrix = [index of nodes, i, j, m, n,
                                  %x1, y1, x2, y2, x3, y3, x4, y4]
end
```

#### 7.5.7.3 *Improved Bilinear Quadratic Element Stiffness*

```
function k =
ImprovedIIBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,y4,
p)
%ImprovedIIBilinearQuadElementStiffness This function returns the
% element stiffness matrix for a bilinear quadrilateral element with
%modulus of elasticity E, Poisson's ratio NU, thickness h, coordinates
%of node 1 (x1,y1), coordinates node 3 (x3,y3), and coordinates of
% node 4 (x4,y4). Use p = 1 for cases of plane stress, and p = 2 for
%cases of plane strain. The size of the element stiffness matrix is 12
x 12.
syms s t

if p == 1
    D = (E/(1-NU*NU))*[1, NU, 0; NU, 1, 0; 0, 0, (1-NU)/2];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0; NU, 1-NU, 0;
    0, 0, (1-2*NU)/2];
end

%Jacobian matrix
J22 = (y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s))/4;
J12 = (y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t))/4;
J11 = (x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t))/4;
J21 = (x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s))/4;
J = (J11*J22) - (J12*J21);

%Compatible displacements
B1c = [J22*(t-1)/4-J12*(s-1)/4 0 ; 0 J11*(s-1)/4-J21*(t-1)/4;
    J11*(s-1)/4-J21*(t-1)/4 J22*(t-1)/4-J12*(s-1)/4];
B2c = [J22*(1-t)/4-J12*(-1-s)/4 0 ; 0 J11*(-1-s)/4-J21*(1-t)/4;
    J11*(-1-s)/4-J21*(1-t)/4 J22*(1-t)/4-J12*(-1-s)/4];
B3c = [J22*(t+1)/4-J12*(s+1)/4 0 ; 0 J11*(s+1)/4-J21*(t+1)/4;
    J11*(s+1)/4-J21*(t+1)/4 J22*(t+1)/4-J12*(s+1)/4];
B4c = [J22*(-1-t)/4-J12*(1-s)/4 0 ; 0 J11*(1-s)/4-J21*(-1-t)/4;
    J11*(1-s)/4-J21*(-1-t)/4 J22*(-1-t)/4-J12*(1-s)/4];
Bfirstc = [B1c B2c B3c B4c];
Bc = Bfirstc/J;
Bcc = J*transpose(Bc)*D*Bc;
kcc = h*int(int(Bcc, t, -1, 1), s, -1, 1);

%Incompatible displacements
B1i = [J22 -J12 0 0; 0 0 -J21 J11; -J21 J11 J22 -J12]/J;
B2i = [-2*s 0 0 0; 0 -2*t 0 0; 0 0 -2*s 0; 0 0 0 -2*t];
Bi = B1i*B2i;
Bicorr = (-1/(h*(x2-x1)*(y3-y2)))*J*h*int(int(Bi, t, -1, 1), s, -1,
1);
```

```
Binew = Bi + Bicorr;
Bii = transpose(Binew)*D*Binew;


Bci = transpose(Bc)*D*Binew;
Bic = transpose(Binew)*D*Bc;

kii = J*h*int(int(Bii, t, -1, 1), s, -1, 1);
kci = J*h*int(int(Bci, t, -1, 1), s, -1, 1);
kic = J*h*int(int(Bic, t, -1, 1), s, -1, 1);

kz = [kcc kci; kic kii];
k = double(kz);
end
```

### 7.5.7.4 *Improved Bilinear Quadratic Assembly*

```
function y = ImprovedBilinearQuadAssemble(K,k,i,j,m,n,ij,jm,mn,ni)
%ImprovedBilinearQuadAssemble This function assembles the element
% stiffness matrix k of the bilinear quadrilateral element with nodes
%i, j, m, n, ij, jm, mn and ni into the global stiffness matrix K.
% This function returns the global stiffness matrix K after the
%element stiffness matrix k is assembled.
K(2*i-1,2*i-1) = K(2*i-1,2*i-1)+k(1,1);
K(2*i-1,2*i) = K(2*i-1,2*i)+k(1,2);
K(2*i-1,2*j-1) = K(2*i-1,2*j-1)+k(1,3);
K(2*i-1,2*j) = K(2*i-1,2*j)+k(1,4);
K(2*i-1,2*m-1) = K(2*i-1,2*m-1)+k(1,5);
K(2*i-1,2*m) = K(2*i-1,2*m)+k(1,6);
K(2*i-1,2*n-1) = K(2*i-1,2*n-1)+k(1,7);
K(2*i-1,2*n) = K(2*i-1,2*n)+k(1,8);
K(2*i-1,ij) = K(2*i-1,ij)+k(1,9);
K(2*i-1,jm) = K(2*i-1,jm)+k(1,10);
K(2*i-1,mn) = K(2*i-1,mn)+k(1,11);
K(2*i-1,ni) = K(2*i-1,ni)+k(1,12);

K(2*i,2*i-1) = K(2*i,2*i-1)+k(2,1);
K(2*i,2*i) = K(2*i,2*i)+k(2,2);
K(2*i,2*j-1) = K(2*i,2*j-1)+k(2,3);
K(2*i,2*j) = K(2*i,2*j)+k(2,4);
K(2*i,2*m-1) = K(2*i,2*m-1)+k(2,5);
K(2*i,2*m) = K(2*i,2*m)+k(2,6);
K(2*i,2*n-1) = K(2*i,2*n-1)+k(2,7);
K(2*i,2*n) = K(2*i,2*n)+k(2,8);
K(2*i,ij) = K(2*i,ij)+k(2,9);
K(2*i,jm) = K(2*i,jm)+k(2,10);
K(2*i,mn) = K(2*i,mn)+k(2,11);
K(2*i,ni) = K(2*i,ni)+k(2,12);

K(2*j-1,2*i-1) = K(2*j-1,2*i-1)+k(3,1);
K(2*j-1,2*i) = K(2*j-1,2*i)+k(3,2);
K(2*j-1,2*j-1) = K(2*j-1,2*j-1)+k(3,3);
K(2*j-1,2*j) = K(2*j-1,2*j)+k(3,4);
K(2*j-1,2*m-1) = K(2*j-1,2*m-1)+k(3,5);
K(2*j-1,2*m) = K(2*j-1,2*m)+k(3,6);
K(2*j-1,2*n-1) = K(2*j-1,2*n-1)+k(3,7);
K(2*j-1,2*n) = K(2*j-1,2*n)+k(3,8);
K(2*j-1,ij) = K(2*j-1,ij)+k(3,9);
K(2*j-1,jm) = K(2*j-1,jm)+k(3,10);
K(2*j-1,mn) = K(2*j-1,mn)+k(3,11);
```

```
K(2*j-1,ni) = K(2*j-1,ni)+k(3,12);

K(2*j,2*i-1) = K(2*j,2*i-1)+k(4,1);
K(2*j,2*i) = K(2*j,2*i)+k(4,2);
K(2*j,2*j-1) = K(2*j,2*j-1)+k(4,3);
K(2*j,2*j) = K(2*j,2*j)+k(4,4);
K(2*j,2*m-1) = K(2*j,2*m-1)+k(4,5);
K(2*j,2*m) = K(2*j,2*m)+k(4,6);
K(2*j,2*n-1) = K(2*j,2*n-1)+k(4,7);
K(2*j,2*n) = K(2*j,2*n)+k(4,8);
K(2*j,ij) = K(2*j,ij)+k(4,9);
K(2*j,jm) = K(2*j,jm)+k(4,10);
K(2*j,mn) = K(2*j,mn)+k(4,11);
K(2*j,ni) = K(2*j,ni)+k(4,12);

K(2*m-1,2*i-1) = K(2*m-1,2*i-1)+k(5,1);
K(2*m-1,2*i) = K(2*m-1,2*i)+k(5,2);
K(2*m-1,2*j-1) = K(2*m-1,2*j-1)+k(5,3);
K(2*m-1,2*j) = K(2*m-1,2*j)+k(5,4);
K(2*m-1,2*m-1) = K(2*m-1,2*m-1)+k(5,5);
K(2*m-1,2*m) = K(2*m-1,2*m)+k(5,6);
K(2*m-1,2*n-1) = K(2*m-1,2*n-1)+k(5,7);
K(2*m-1,2*n) = K(2*m-1,2*n)+k(5,8);
K(2*m-1,ij) = K(2*m-1,ij)+k(5,9);
K(2*m-1,jm) = K(2*m-1,jm)+k(5,10);
K(2*m-1,mn) = K(2*m-1,mn)+k(5,11);
K(2*m-1,ni) = K(2*m-1,ni)+k(5,12);

K(2*m,2*i-1) = K(2*m,2*i-1)+k(6,1);
K(2*m,2*i) = K(2*m,2*i)+k(6,2);
K(2*m,2*j-1) = K(2*m,2*j-1)+k(6,3);
K(2*m,2*j) = K(2*m,2*j)+k(6,4);
K(2*m,2*m-1) = K(2*m,2*m-1)+k(6,5);
K(2*m,2*m) = K(2*m,2*m)+k(6,6);
K(2*m,2*n-1) = K(2*m,2*n-1)+k(6,7);
K(2*m,2*n) = K(2*m,2*n)+k(6,8);
K(2*m,ij) = K(2*m,ij)+k(6,9);
K(2*m,jm) = K(2*m,jm)+k(6,10);
K(2*m,mn) = K(2*m,mn)+k(6,11);
K(2*m,ni) = K(2*m,ni)+k(6,12);

K(2*n-1,2*i-1) = K(2*n-1,2*i-1)+k(7,1);
K(2*n-1,2*i) = K(2*n-1,2*i)+k(7,2);
K(2*n-1,2*j-1) = K(2*n-1,2*j-1)+k(7,3);
K(2*n-1,2*j) = K(2*n-1,2*j)+k(7,4);
K(2*n-1,2*m-1) = K(2*n-1,2*m-1)+k(7,5);
K(2*n-1,2*m) = K(2*n-1,2*m)+k(7,6);
K(2*n-1,2*n-1) = K(2*n-1,2*n-1)+k(7,7);
K(2*n-1,2*n) = K(2*n-1,2*n)+k(7,8);
K(2*n-1,ij) = K(2*n-1,ij)+k(7,9);
K(2*n-1,jm) = K(2*n-1,jm)+k(7,10);
K(2*n-1,mn) = K(2*n-1,mn)+k(7,11);
K(2*n-1,ni) = K(2*n-1,ni)+k(7,12);

K(2*n,2*i-1) = K(2*n,2*i-1)+k(8,1);
K(2*n,2*i) = K(2*n,2*i)+k(8,2);
K(2*n,2*j-1) = K(2*n,2*j-1)+k(8,3);
K(2*n,2*j) = K(2*n,2*j)+k(8,4);
K(2*n,2*m-1) = K(2*n,2*m-1)+k(8,5);
K(2*n,2*m) = K(2*n,2*m)+k(8,6);
K(2*n,2*n-1) = K(2*n,2*n-1)+k(8,7);
K(2*n,2*n) = K(2*n,2*n)+k(8,8);
K(2*n,ij) = K(2*n,ij)+k(8,9);
```

```
K(2*n,jm) = K(2*n,jm)+k(8,10);
K(2*n,mn) = K(2*n,mn)+k(8,11);
K(2*n,ni) = K(2*n,ni)+k(8,12);

K(ij,2*i-1) = K(ij,2*i-1)+k(9,1);
K(ij,2*i) = K(ij,2*i)+k(9,2);
K(ij,2*j-1) = K(ij,2*j-1)+k(9,3);
K(ij,2*j) = K(ij,2*j)+k(9,4);
K(ij,2*m-1) = K(ij,2*m-1)+k(9,5);
K(ij,2*m) = K(ij,2*m)+k(9,6);
K(ij,2*n-1) = K(ij,2*n-1)+k(9,7);
K(ij,2*n) = K(ij,2*n)+k(9,8);
K(ij,ij) = K(ij,ij)+k(9,9);
K(ij,jm) = K(ij,jm)+k(9,10);
K(ij,mn) = K(ij,mn)+k(9,11);
K(ij,ni) = K(ij,ni)+k(9,12);

K(jm,2*i-1) = K(jm,2*i-1)+k(10,1);
K(jm,2*i) = K(jm,2*i)+k(10,2);
K(jm,2*j-1) = K(jm,2*j-1)+k(10,3);
K(jm,2*j) = K(jm,2*j)+k(10,4);
K(jm,2*m-1) = K(jm,2*m-1)+k(10,5);
K(jm,2*m) = K(jm,2*m)+k(10,6);
K(jm,2*n-1) = K(jm,2*n-1)+k(10,7);
K(jm,2*n) = K(jm,2*n)+k(10,8);
K(jm,ij) = K(jm,ij)+k(10,9);
K(jm,jm) = K(jm,jm)+k(10,10);
K(jm,mn) = K(jm,mn)+k(10,11);
K(jm,ni) = K(jm,ni)+k(10,12);

K(mn,2*i-1) = K(mn,2*i-1)+k(11,1);
K(mn,2*i) = K(mn,2*i)+k(11,2);
K(mn,2*j-1) = K(mn,2*j-1)+k(11,3);
K(mn,2*j) = K(mn,2*j)+k(11,4);
K(mn,2*m-1) = K(mn,2*m-1)+k(11,5);
K(mn,2*m) = K(mn,2*m)+k(11,6);
K(mn,2*n-1) = K(mn,2*n-1)+k(11,7);
K(mn,2*n) = K(mn,2*n)+k(11,8);
K(mn,ij) = K(mn,ij)+k(11,9);
K(mn,jm) = K(mn,jm)+k(11,10);
K(mn,mn) = K(mn,mn)+k(11,11);
K(mn,ni) = K(mn,ni)+k(11,12);

K(ni,2*i-1) = K(ni,2*i-1)+k(12,1);
K(ni,2*i) = K(ni,2*i)+k(12,2);
K(ni,2*j-1) = K(ni,2*j-1)+k(12,3);
K(ni,2*j) = K(ni,2*j)+k(12,4);
K(ni,2*m-1) = K(ni,2*m-1)+k(12,5);
K(ni,2*m) = K(ni,2*m)+k(12,6);
K(ni,2*n-1) = K(ni,2*n-1)+k(12,7);
K(ni,2*n) = K(ni,2*n)+k(12,8);
K(ni,ij) = K(ni,ij)+k(12,9);
K(ni,jm) = K(ni,jm)+k(12,10);
K(ni,mn) = K(ni,mn)+k(12,11);
K(ni,ni) = K(ni,ni)+k(12,12);

y = K;
end
```

### 7.5.7.5   *Stress Calculation Q6*

```
function [ sigma,sigmaX, sigmaBX, xB, yB, Qcentroid, Qtop ] =
StressCalculationQ6( E,NU,U,Lx,Ly,dx,dy,element,p,xe,ye )
%StressDistribution     This function returns a graph between the
%stress values of the beam and its length. E is the modulus of
%elasticity, NU is Poisson's ratio, U is the displacement vector of
%the beam, Lx is the Length of the beam, Ly is the beam's height, dx
%is the length of the mesh in x-direction, dy is the length of the
%mesh in y-direction, element is a matrix contains information about
%the nodes and their coordinates, p is the problem that is going to be
%uses (1 for stress and 2 for strain) and finally xe and ye are the
%coordinates of the point that the stresses are going to be
%calculated.
nx = Lx/dx;
ny = Ly/dy;
no = 0;
for q = 1: size(element,1)
    a = element(q,2);
    b = element(q,3);
    c = element(q,4);
    d = element(q,5);
    ab = element(q,6);
    bc = element(q,7);
    cd = element(q,8);
    da = element(q,9);
    x1 = element(q,10);
    y1 = element(q,11);
    x2 = element(q,12);
    y2 = element(q,13);
    x3 = element(q,14);
    y3 = element(q,15);
    x4 = element(q,16);
    y4 = element(q,17);
    uu((12*no)+1:(12*no)+12,1)=[U((((a*2)-1)):(((a*2)+2)));
     U(((c*2)-1));U((c*2));U(((d*2)-
     1));U((d*2));U(ab,1);U(bc,1);U(cd,1);U(da,1)];

    %sigma is the stresses at xe = [-dx/2 and dx/2] & ye = 0 to draw a
    %distribution of the normal stresses along the beam and compare it
    %to the exact ones
     sigma((3*no)+1:(3*no)+3,:) =
     ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
     ,p,uu((12*no)+1:(12*no)+12,1),xe,ye);
    %to calculate shear stress
     sigmaQcentroid((3*no)+1:(3*no)+3,:) =
     ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
     ,p,uu((12*no)+1:(12*no)+12,1),xe,0);
    %sigma0 is the stresses at the centroid of the element
     sigma0((3*no)+1:(3*no)+3,:) =
     ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
     ,p,uu((12*no)+1:(12*no)+12,1),0,0);
    %sigmabottom is the stresses of all elements at their bottom side
     sigmabottom((3*no)+1:(3*no)+3,:) =
     ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
     ,p,uu((12*no)+1:(12*no)+12,1),-dx/2,-dy/2);
    %sigmatop is the stesses of all elements at their top right side
     sigmatop((3*no)+1:(3*no)+3,:) =
     ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
     ,p,uu((12*no)+1:(12*no)+12,1),dx/2,dy/2);
    %sigmaleft is the stresses of all elements at their top left side
    sigmaleft((3*no)+1:(3*no)+3,:) =
    ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
    ,p,uu((12*no)+1:(12*no)+12,1),-dx/2,dy/2);
    %sigma right is the stresses of all elements at their bottom right
    side
    sigmaright((3*no)+1:(3*no)+3,:) =
    ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4
    ,p,uu((12*no)+1:(12*no)+12,1),dx/2,-dy/2);
    no = no+1;
end

%n are the number of elements used
n = size(element,1);

%sigmaBX are the normal stresses at the boundary nodes of the beams
sigma0X = sigma0(1:3:size(sigma0),:);
sigmatopX = sigmatop(1:3:size(sigmatop),:);
sigmabottomX = sigmabottom(1:3:size(sigmabottom),:);
sigmaleftX = sigmaleft(1:3:size(sigmaleft),:);
sigmarightX = sigmaright(1:3:size(sigmaright),:);

x1 = element(:,10);
y1 = element(:,11);
x2 = element(:,12);
y2 = element(:,13);
x3 = element(:,14);
y3 = element(:,15);
x4 = element(:,16);
y4 = element(:,17);
sigmaBbottomX = sigmabottomX(1:nx,:);
xbottom = x1(1:nx,:);
ybottom = y1(1:nx,:);
sigmaBtopX = sigmatopX(n-nx+1:n,:);
xtop = x3(n-nx+1:n,:);
ytop = y3(n-nx+1:n,:);
sigmaBleftX = sigmaleftX(1:nx:n,:);
xleft = x4(1:nx:n,:);
yleft = y4(1:nx:n,:);
sigmaBrightX = sigmarightX(nx:nx:n,:);
xright = x2(nx:nx:n,:);
yright = y2(nx:nx:n,:);

sigmaBX = [sigma0X; sigmaBbottomX; sigmaBtopX; sigmaBleftX;
sigmaBrightX];
xB = [xbottom; xtop; xleft; xright];
yB = [ybottom; ytop; yleft; yright];

%sigmaX are the normal stresses
sigmaX = sigma(1:3:size(sigma),:);

%Q is the shear stress
Qcentroid = sigmaQcentroid(3:3:size(sigmaQcentroid),:);
Qtop = sigma(3:3:size(sigma),:);
end
```

#### *7.5.7.6 Improved Bilinear Quadratic Element Stresses*

```
function w =
ImprovedIIBilinearQuadElementStresses(E,NU,x1,y1,x2,y2,x3,y3,x4,y4,p,u
,xe,ye)
```

```
%ImprovedIIBilinearQuadElementStresses This function returns the
%element stress vector for incompatible element with modulus of
%elasticity E, Poisson’s ratio NU, coordinates of node 1 (x1,y1),
%coordinates of node 2 (x2,y2), coordinates of node 3 (x3,y3),
%coordinates of node 4 (x4,y4), and element displacement vector u.
% Use p = 1 for cases of plane stress, and p = 2 for cases of plane
strain.
syms s t
if p == 1
    D = (E/(1-NU*NU))*[1, NU, 0; NU, 1, 0; 0, 0, (1-NU)/2];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0; NU, 1-NU, 0;
    0, 0, (1-2*NU)/2];
end



%Compatible displacements
J22 = (y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s))/4;
J12 = (y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t))/4;
J11 = (x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t))/4;
J21 = (x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s))/4;
B1c = [J22*(t-1)/4-J12*(s-1)/4 0 ; 0 J11*(s-1)/4-J21*(t-1)/4;
    J11*(s-1)/4-J21*(t-1)/4 J22*(t-1)/4-J12*(s-1)/4];
B2c = [J22*(1-t)/4-J12*(-1-s)/4 0 ; 0 J11*(-1-s)/4-J21*(1-t)/4;
    J11*(-1-s)/4-J21*(1-t)/4 J22*(1-t)/4-J12*(-1-s)/4];
B3c = [J22*(t+1)/4-J12*(s+1)/4 0 ; 0 J11*(s+1)/4-J21*(t+1)/4;
    J11*(s+1)/4-J21*(t+1)/4 J22*(t+1)/4-J12*(s+1)/4];
B4c = [J22*(-1-t)/4-J12*(1-s)/4 0 ; 0 J11*(1-s)/4-J21*(-1-t)/4;
    J11*(1-s)/4-J21*(-1-t)/4 J22*(-1-t)/4-J12*(1-s)/4];
Bfirstc = [B1c B2c B3c B4c];
J = (J11*J22) - (J12*J21);
Bc = Bfirstc/J;

%Incompatible displacements
B1i = [J22 -J12 0 0; 0 0 -J21 J11; -J21 J11 J22 -J12]/J;
B2i = [-2*s 0 0 0; 0 -2*t 0 0; 0 0 -2*s 0; 0 0 0 -2*t];
Bi = B1i*B2i;
Bicorr = (-1/((x2-x1)*(y3-y2)))*J*int(int(Bi, t, -1, 1), s, -1, 1);
Binew = Bi + Bicorr;
B = [Bc Binew];

w = D*B*u;
% to calculate the stresses at the centroid of the element
wcent = subs(w, {s,t}, {xe,ye});
w = double(wcent);
end
```

### *7.5.7.7 Draw Beam Q6*

```
function y = DrawBeamQ6( element )
%DrawBeam This function returns a graph for an undeformed beam after
meshing with incompatible elements. The input is element matrix.

hold on
for ii = 1: size(element,1)
    x1 = element(ii,10);
    y1 = element(ii,11);
    x2 = element(ii,12);
    y2 = element(ii,13);
    x3 = element(ii,14);
    y3 = element(ii,15);
    x4 = element(ii,16);
    y4 = element(ii,17);
    plot([x1, x2], [y1, y2],'color',[1 0 0]);
    plot([x2, x3], [y2, y3],'color',[1 0 0]);
    plot([x3, x4], [y3, y4],'color',[1 0 0]);
    plot([x4, x1], [y4, y1],'color',[1 0 0]);
end
hold off
end
```

### 7.5.7.8 *Draw Deformed Beam Q6*

```
function [ ] = DrawDeformedBeamQ6( element,U )
%DrawDeformedBeam This function returns a graph for a beam modeled by
%Q8 element after deformation.
%  The input is element matrix and Displacement vector U. (It is
%better to multiply U vector by certain value for better graph.

hold on
axis equal
for ii = 1: size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    ij = element(ii,6);
    jm = element(ii,7);
    mn = element(ii,8);
    ni = element(ii,9);
    x1 = element(ii,10);
    y1 = element(ii,11);
    x2 = element(ii,12);
    y2 = element(ii,13);
    x3 = element(ii,14);
    y3 = element(ii,15);
    x4 = element(ii,16);
    y4 = element(ii,17);
    x5 = (x1 + x2)/2;
    x6 = (x2 + x3)/2;
    x7 = (x3 + x4)/2;
    x8 = (x4 + x1)/2;
    y5 = (y1 + y2)/2;
    y6 = (y2 + y3)/2;
    y7 = (y3 + y4)/2;
    y8 = (y4 + y1)/2;
    ux1 = U((i*2)-1,1);
    ux2 = U((j*2)-1,1);
    ux3 = U((m*2)-1,1);
    ux4 = U((n*2)-1,1);
    uy1 = U((i*2),1);
    uy2 = U((j*2),1);
    uy3 = U((m*2),1);
    uy4 = U((n*2),1);
    ux5 = U(ij,1);
    ux6 = U(ni,1);
    uy5 = U(jm,1);
    uy6 = U(mn,1);
    x1new = x1+ux1;
    x2new = x2+ux2;
    x3new = x3+ux3;
    x4new = x4+ux4;
    y1new = y1+uy1;
    y2new = y2+uy2;
    y3new = y3+uy3;
    y4new = y4+uy4;
    x5new = (x2new+x1new)/2+ux5;
    y5new = (y2new+y1new)/2+uy5;
    x6new = (x3new+x2new)/2+ux6;
    y6new = (y3new+y2new)/2+uy6;
    x7new = (x3new+x4new)/2+ux5;
    y7new = (y3new+y4new)/2+uy5;
    x8new = (x4new+x1new)/2+ux6;
    y8new = (y4new+y1new)/2+uy6;
    centroidnew(ii,:) = [(((x2new+x1new)/2)+((x3new...
        +x4new)/2))/2,(((y3new+y2new)/2)+((y1new...
        +y4new)/2))/2];
    plot([x1new, x5new], [y1new, y5new],':','color',[0 1 0]);
    plot([x5new, x2new], [y5new, y2new],':','color',[0 1 0]);
    plot([x2new, x6new], [y2new, y6new],':','color',[0 1 0]);
    plot([x6new, x3new], [y6new, y3new],':','color',[0 1 0]);
    plot([x3new, x7new], [y3new, y7new],':','color',[0 1 0]);
    plot([x7new, x4new], [y7new, y4new],':','color',[0 1 0]);
    plot([x4new, x8new], [y4new, y8new],':','color',[0 1 0]);
    plot([x8new, x1new], [y8new, y1new],':','color',[0 1 0]);
end
axis tight
xlabel('Length of beam (m)')
ylabel('Depth of beam (m)')
title('Deformed shape of the beam
Q6','FontSize',12,'FontWeight','Bold')
hold off
end
```

## 7.6 Appendix F: Weighted Integration

### *7.6.1 Cantilever and Simple Beams*

Both Cantilever and Simple beams are done as their corresponding in section 7.4 except the line of formulating the stiffness matrix is changed by the following one:

```
[element,K,centroid] =
GaussRemedyBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1,d
y/dx);
```

More details about this line shown in section 7.6.2.

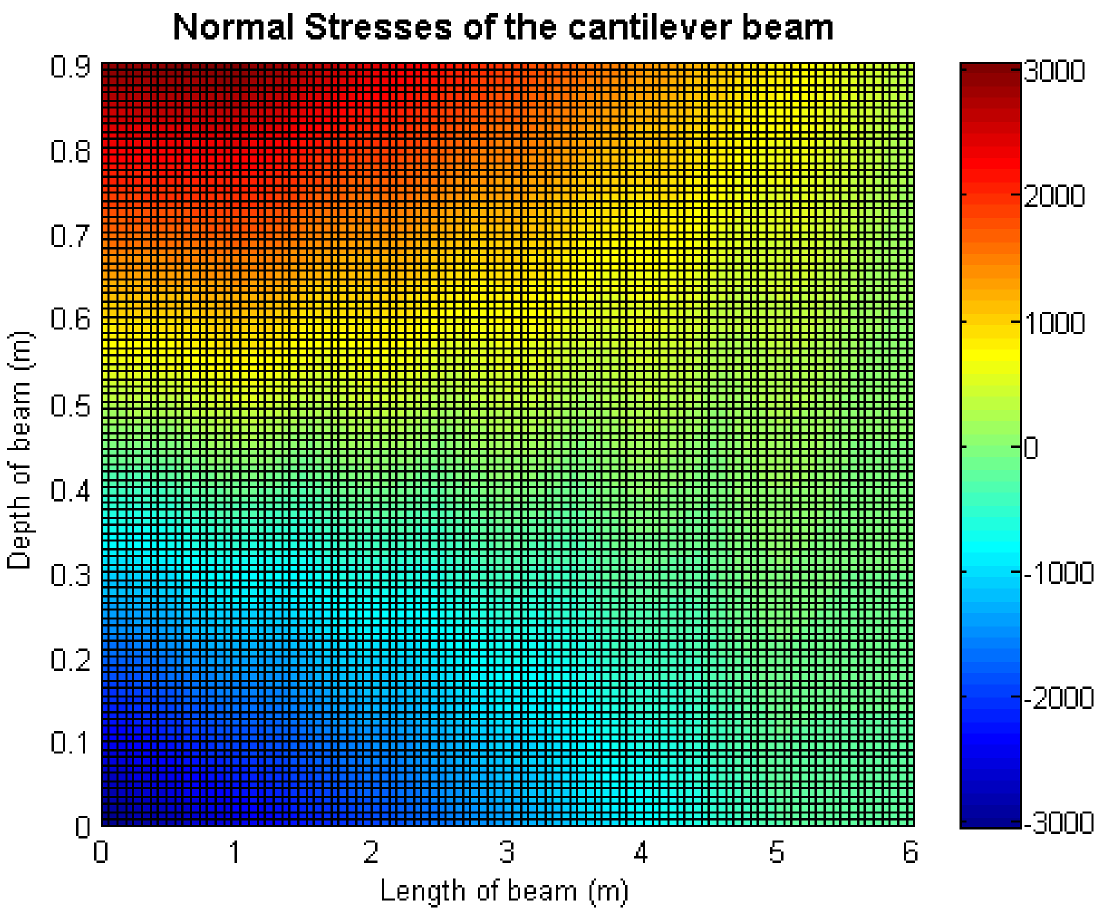


**Fig. 7-19** Analysis of CantileverBeam_C2_01_03_WeightedInt.

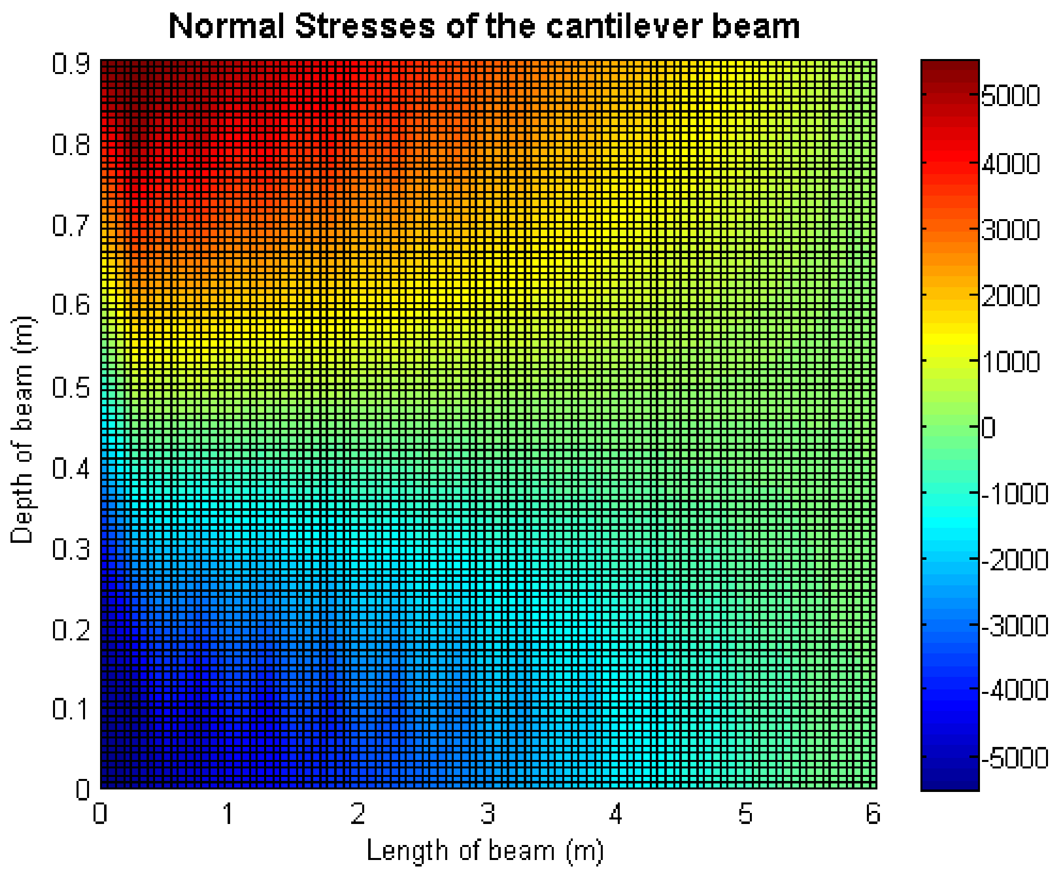


**Fig. 7-20** Analysis of CantileverBeam_C2_02_03_WeightedInt.

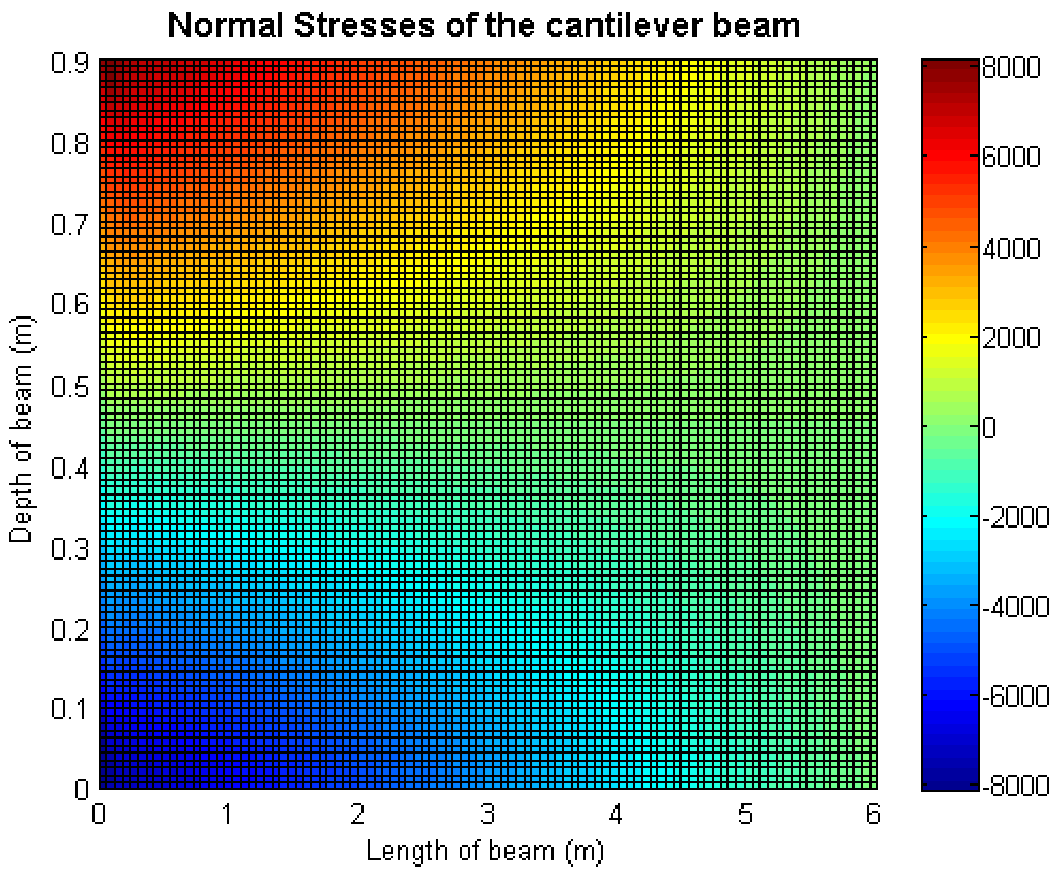
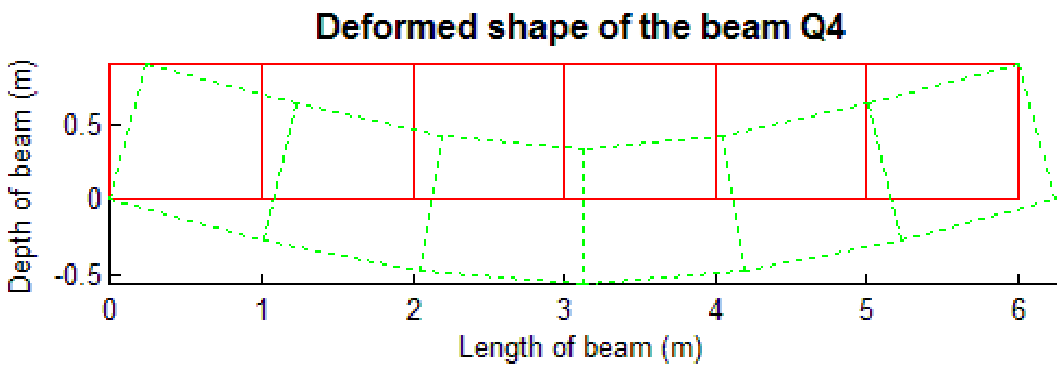
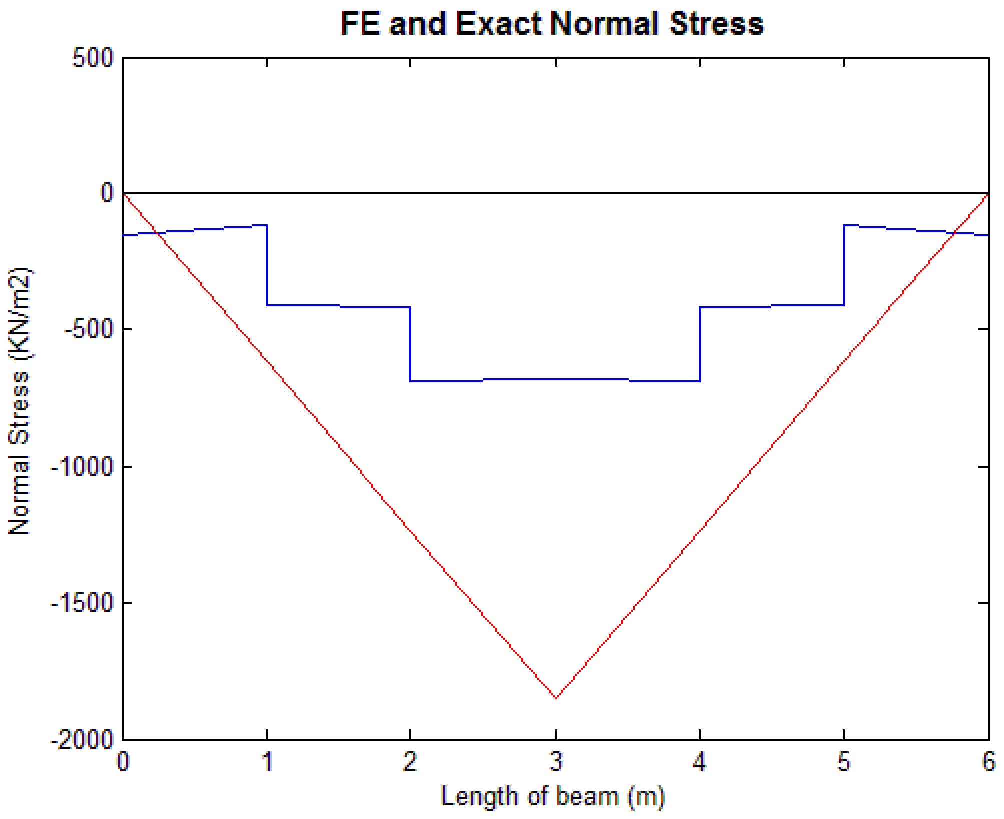


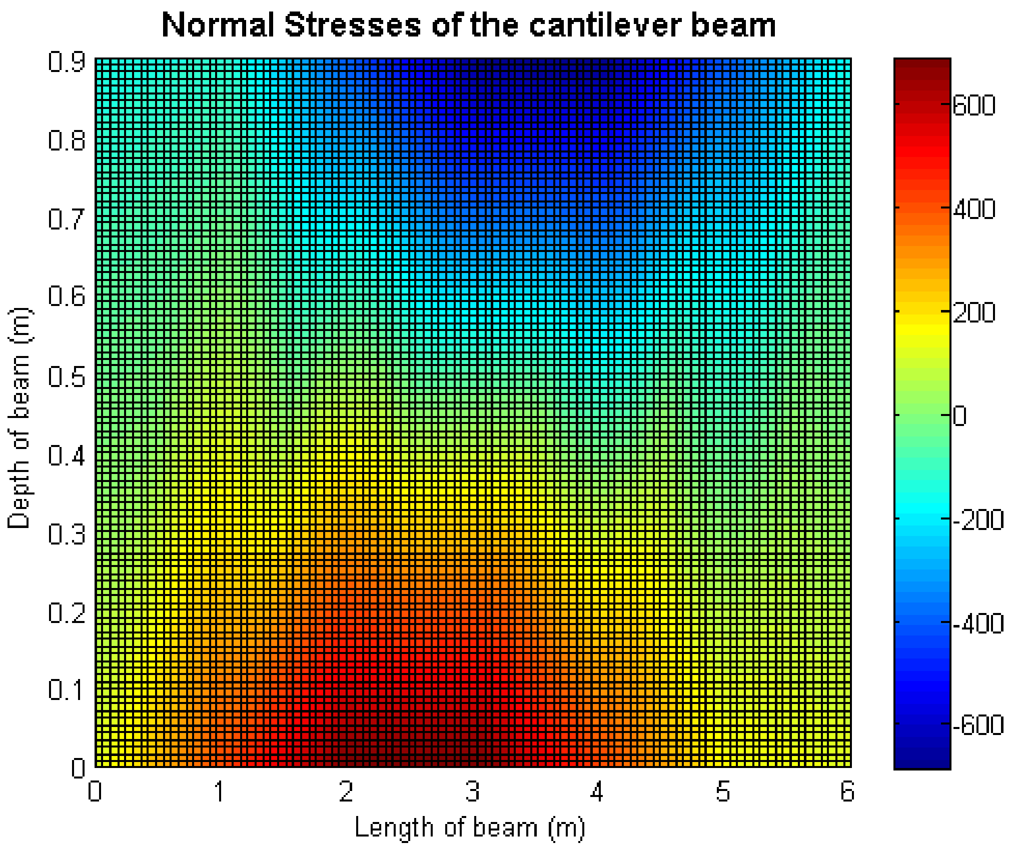
**Fig. 7-21** Analysis ofCantileverBeam_C2_03_03_WeightedInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

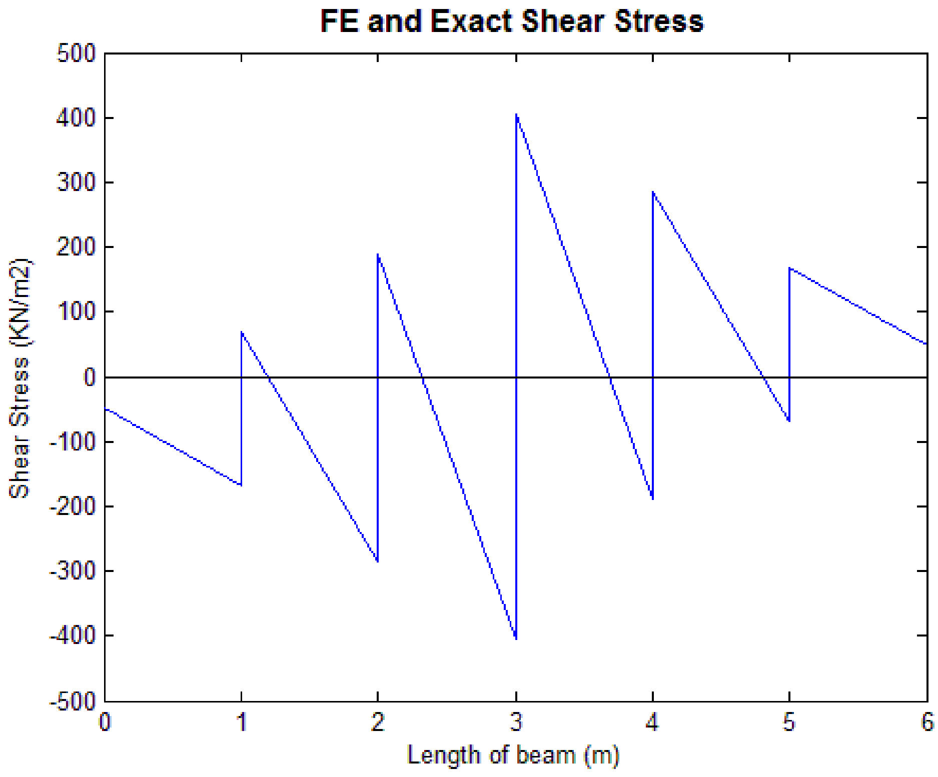
(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6
Length of beam (m)

(b)

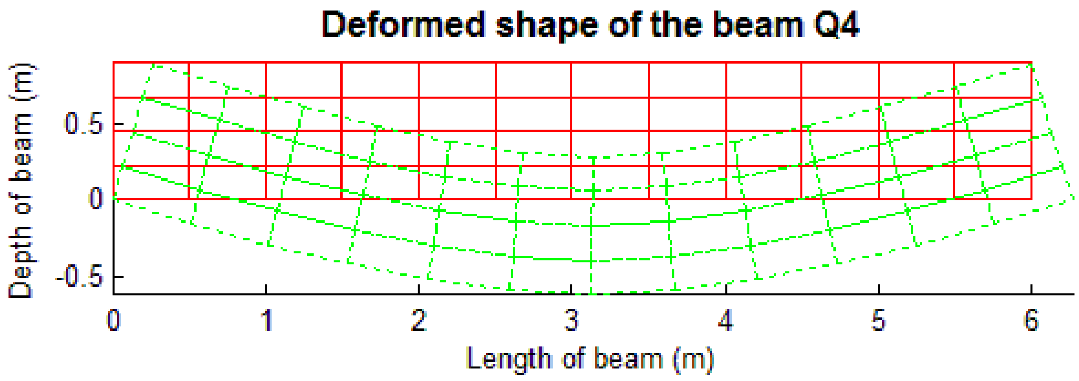


(c)

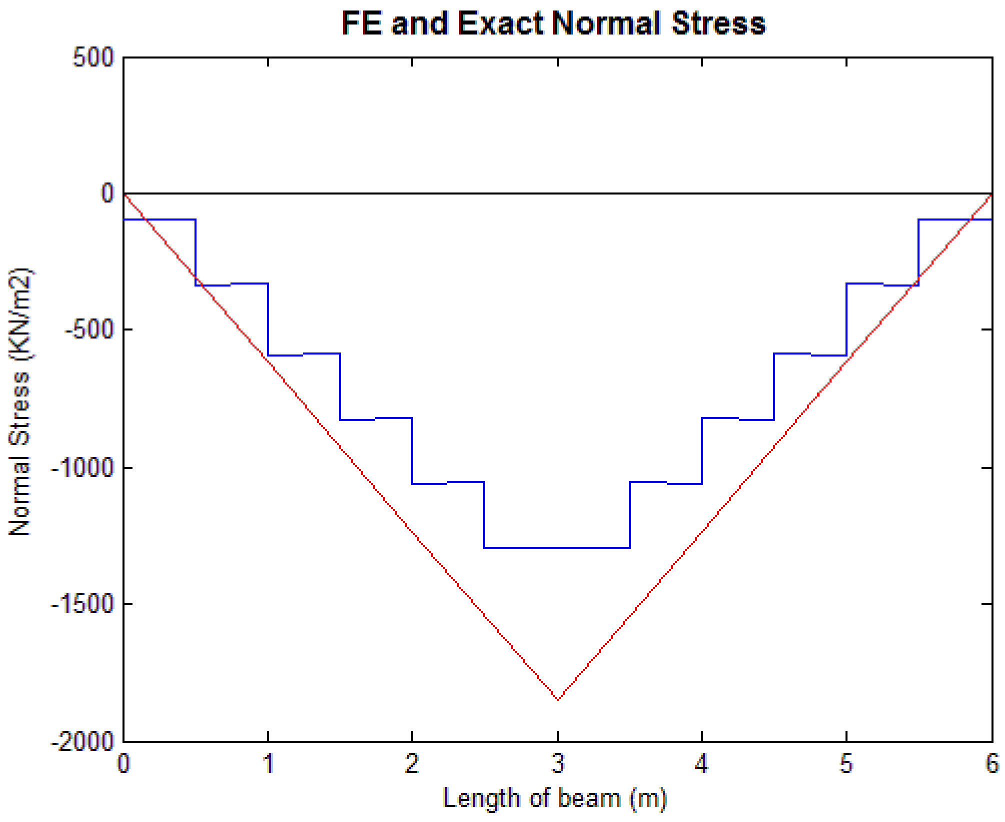


(d)

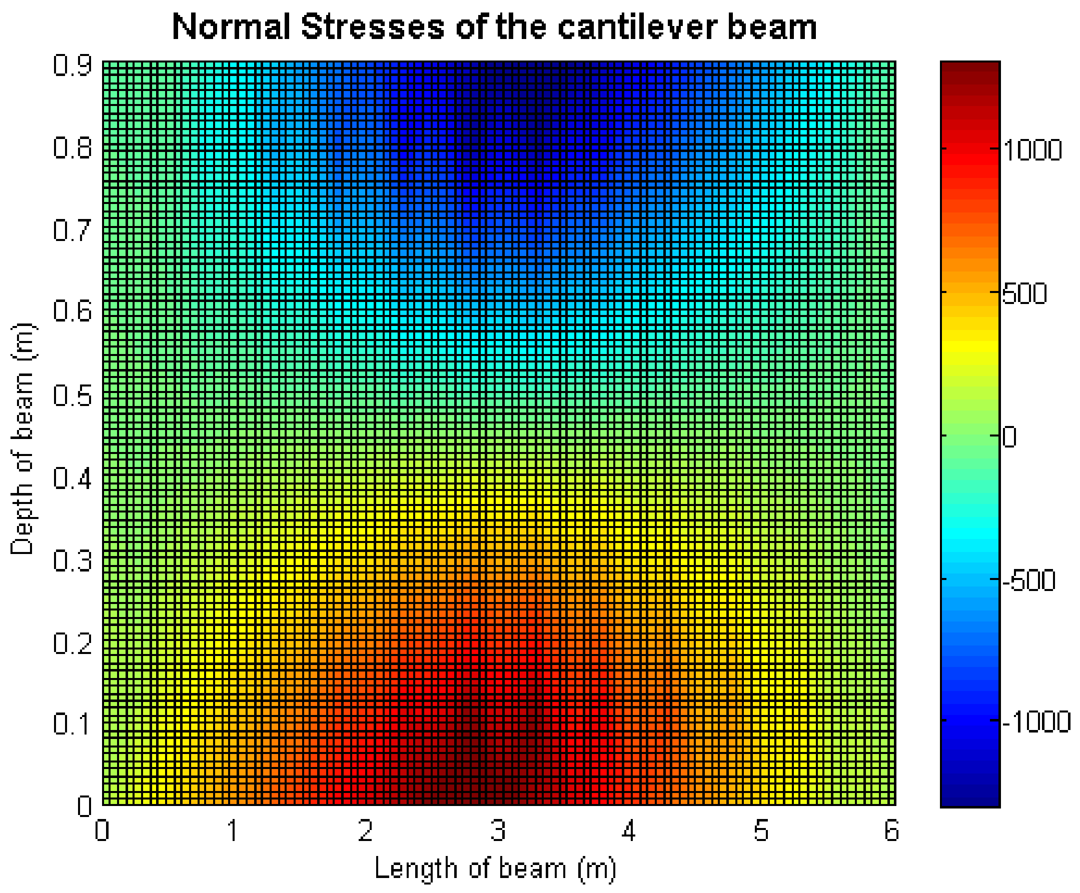
**Fig. 7-22** Analysis of SimpleBeam_A2_01_03_WeightedInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

(a)


FE and Exact Normal Stress
500
0
-500
-1000
-1500
-2000
Normal Stress (KN/m2)
0
1
2
3
4
5
6
Length of beam (m)

(b)



(c)

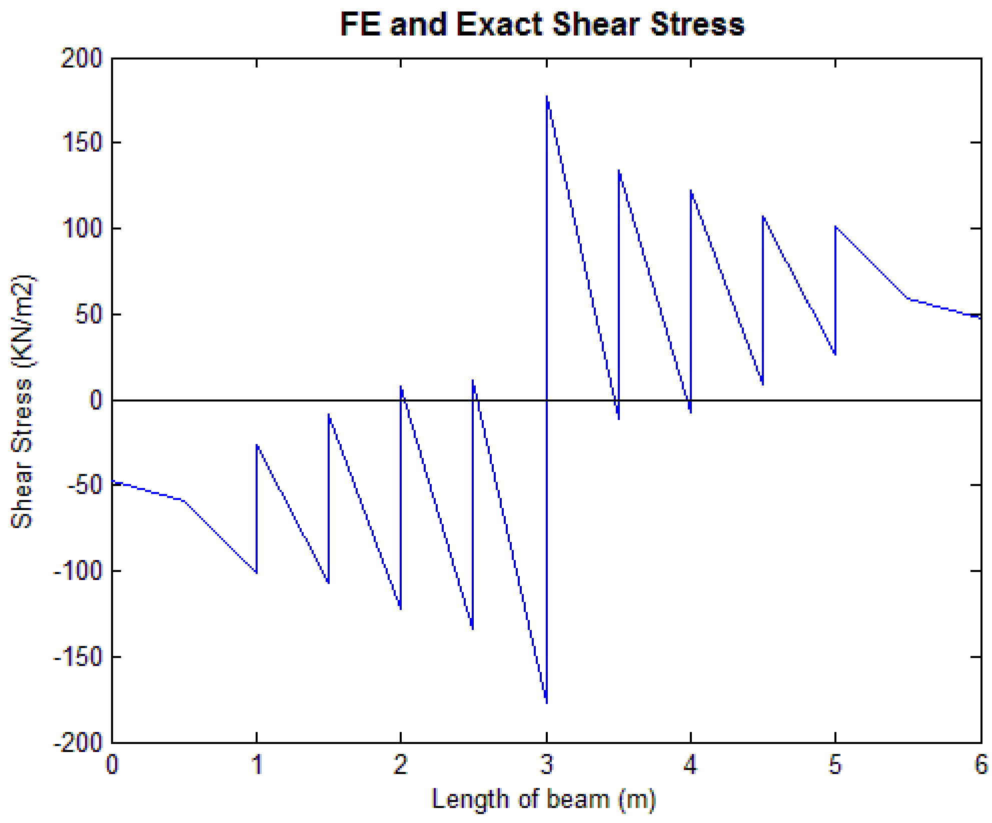
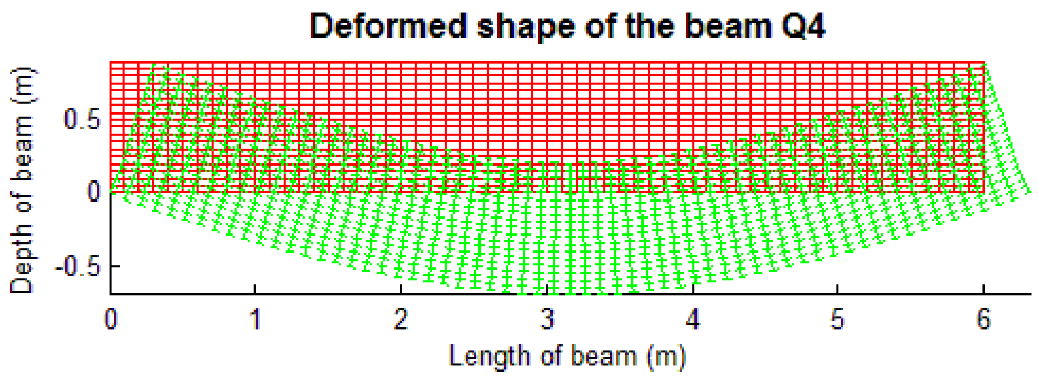
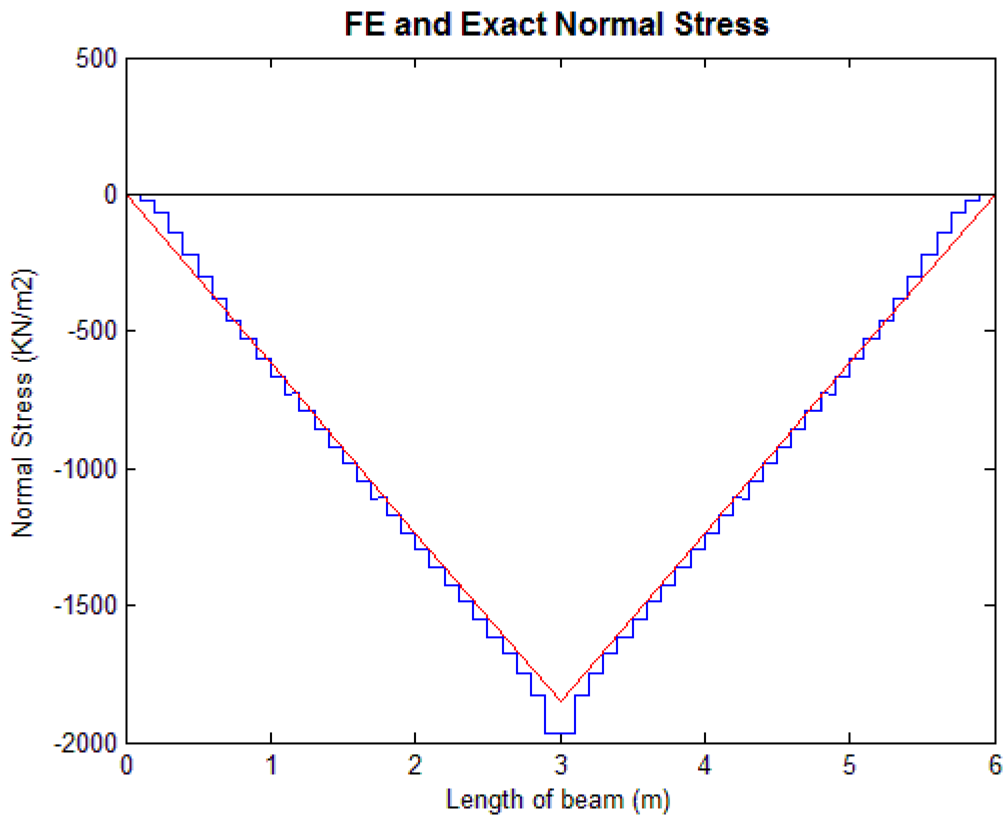


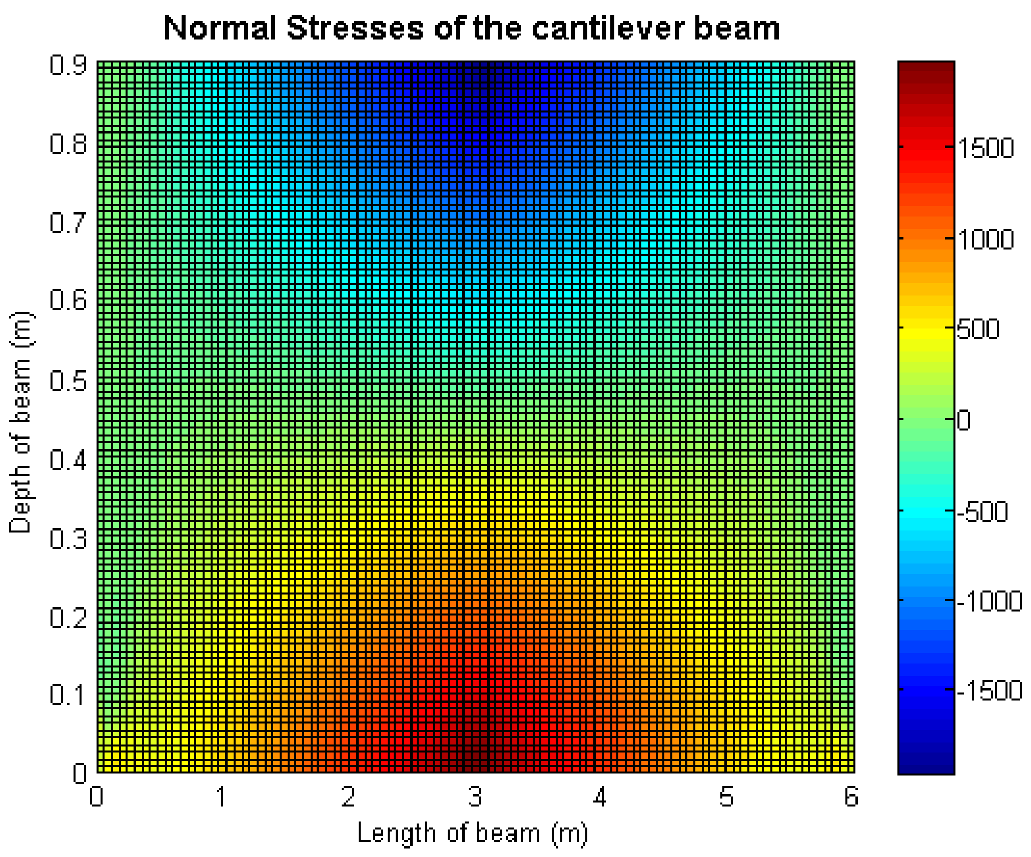
**Fig. 7-23** SimpleBeam_A2_02_03_WeightedInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

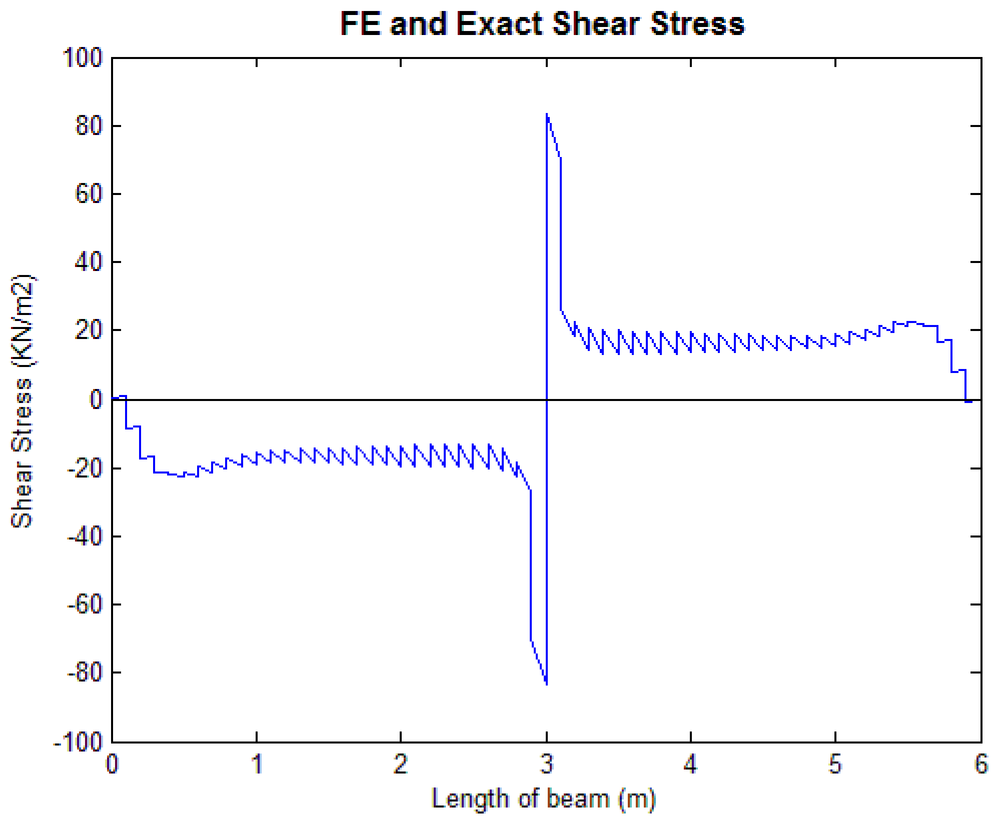
(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6
Length of beam (m)

(b)



(c)



(d)

**Fig. 7-24** Analysis of SimpleBeam_A2_03_03_WeightedInt

### *7.6.2 Other codes*

#### *7.6.2.1 Gauss Remedy Bilinear Quadratic Element Stiffness Assembly*

```
function [element,K,centroid] =
GaussRemedyBilinearQuadElementStiffnessAssembly(E,NU,h,Lx,Ly,dx,dy,p,g
amma)
%GaussRemedyQuadElementStiffnessAssembly This code returns the Global
%stiffness matrix for a beam/plate to be analyzed as 4-noded element
%using Gauss Quadrature.
%   The inputs are modulus of elasticity E, Poisson's ratio NU,
%thickness of the element t, length of the beam Lx, its depth Ly,
%length of element in x dx, length of the element in y Ly, p which is
%1 in case of plain stress problems and 2 in case of plain strain
%problems and gamma which is the aspect ratio. The output is the
%element matrix which contains all the details of each element for the
%beam, the whole global stiffness matrix for the whole beam K and the
%a matrix which contains the positions of the centroid of the elements
%centroid.
[element,nx,ny] = BeamMeshingQ4(Lx,Ly,dx,dy);

no = 0;
K = zeros(2*(nx+1)*(ny+1),2*(nx+1)*(ny+1));
for ii=1:size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    x1 = element(ii,6);
    y1 = element(ii,7);
    x2 = element(ii,8);
    y2 = element(ii,9);
    x3 = element(ii,10);
    y3 = element(ii,11);
    x4 = element(ii,12);
    y4 = element(ii,13);
    centroid(ii,:) = [(x2+x1)/2,(y3+y2)/2];
     k =
     GaussRemedyBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x
     4,y4,p,gamma);
    no = no + 1;
    K = BilinearQuadAssemble(K,k,i,j,m,n);
end
```

#### *7.6.2.2 Gauss Remedy Bilinear Quadratic Element Stiffness*

```
function k =
GaussRemedyBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,y4
,p,gamma)
%GaussRemedyBilinearQuadElementStiffness This function returns the
%element stiffness matrix for a bilinear quadrilateral element with
%modulus of elasticity E, Poisson’s ratio NU, thickness h, coordinates
%of node 1 (x1,y1), coordinates node 3 (x3,y3), and coordinates of
%node 4 (x4,y4) and gamma which is the aspect ratio. Use p = 1 for
%cases of plane stress, and p = 2 for cases of plane strain.
% The size of the element stiffness matrix is 8 x 8.
% The integration is done using Gauss quadrature.
```

```
syms s t;
a = (y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s))/4;
b = (y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t))/4;
c = (x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t))/4;
d = (x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s))/4;
B1 = [a*(t-1)/4-b*(s-1)/4 0 ; 0 c*(s-1)/4-d*(t-1)/4
    c*(s-1)/4-d*(t-1)/4 a*(t-1)/4-b*(s-1)/4];
B2 = [a*(1-t)/4-b*(-1-s)/4 0 ; 0 c*(-1-s)/4-d*(1-t)/4
    c*(-1-s)/4-d*(1-t)/4 a*(1-t)/4-b*(-1-s)/4];
B3 = [a*(t+1)/4-b*(s+1)/4 0 ; 0 c*(s+1)/4-d*(t+1)/4
    c*(s+1)/4-d*(t+1)/4 a*(t+1)/4-b*(s+1)/4];
B4 = [a*(-1-t)/4-b*(1-s)/4 0 ; 0 c*(1-s)/4-d*(-1-t)/4
    c*(1-s)/4-d*(-1-t)/4 a*(-1-t)/4-b*(1-s)/4];
Bfirst = [B1 B2 B3 B4];
Jfirst = [0 1-t t-s s-1 ; t-1 0 s+1 -s-t ;
    s-t -s-1 0 t+1 ; 1-s s+t -t-1 0];
J = [x1 x2 x3 x4]*Jfirst*[y1 ; y2 ; y3 ; y4]/8;
B = Bfirst/J;
if p == 1
    D = (E/(1-NU*NU))*[1, NU, 0; NU, 1, 0; 0, 0, (1-NU)/2];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0; NU, 1-NU, 0;
    0, 0, (1-2*NU)/2];
end
BD = J*transpose(B)*D*B;
%Integration of stiffness matrix using 1 gauss point
k1x1 = intGQ(1,BD);
%Integration of stiffness matrix using 2 gauss points
k2x2 = intGQ(2,BD);
%factor to adjust shear locking
beta = ((2/gamma^2)*(1-NU^2))/(1+(2/(gamma^2))-NU);
%Stiffness matrix by weighted integration method
rB = ((1-beta)*k1x1)+(beta*k2x2);
z = h*rB;
k = double(z);
end
```

#### *7.6.2.3 Gauss Integration*

```
function [ I ] = intGQ( n,fx )
%intQD This function solves the integration using Gauss Quadrature
method.
%   n is the number of Gauss points used in the integration and fx in
the function to be integrated. This code is up to 4 Gauss points.
syms s t
if n == 1
    I = 2*subs(subs(fx,s,0),t,0);
    I = double(I);
else
if n == 2
    w1 = 1;
    ss1 = -sqrt(3)/3;
    tt1 = -sqrt(3)/3;
    ss2 = sqrt(3)/3;
    tt2 = sqrt(3)/3;
    I = w1*subs(subs(fx,s,ss1),t,tt1)+w1*subs(subs(fx,s,ss2),t,tt1)...
        +w1*subs(subs(fx,s,ss1),t,tt2)+w1*subs(subs(fx,s,ss2),t,tt2);
    I = double(I);
else
    if n == 3
        w1 = 5/9;
        w2 = 8/9;
        w3 = 5/9;
        ss1 = -sqrt(0.6);
        tt1 = -sqrt(0.6);
        ss2 = 0;
        tt2 = 0;
        ss3 = sqrt(0.6);
        tt3 = sqrt(0.6);
        I = w1*[w1*subs(subs(fx,s,ss1),t,tt1) + ...
                w2*subs(subs(fx,s,ss2),t,tt1) + ...
                w3*subs(subs(fx,s,ss3),t,tt1)] + ...
            w2*[w1*subs(subs(fx,s,ss1),t,tt2) + ...
                w2*subs(subs(fx,s,ss2),t,tt2) + ...
                w3*subs(subs(fx,s,ss3),t,tt2)] + ...
            w3*[w1*subs(subs(fx,s,ss1),t,tt3) + ...
                w2*subs(subs(fx,s,ss2),t,tt3) + ...
                w3*subs(subs(fx,s,ss3),t,tt3)];
        I = double(I);
    else
        if n == 4
            w1 = 0.6521451548625461;
            w2 = 0.6521451548625461;
            w3 = 0.3478548451374538;
            w4 = 0.3478548451374538;
            ss1 = -0.3399810435848563;
            tt1 = -0.3399810435848563;
            ss2 = 0.3399810435848563;
            tt2 = 0.3399810435848563;
            ss3 = -0.8611363115940526;
            tt3 = -0.8611363115940526;
            ss4 = 0.8611363115940526;
            tt4 = 0.8611363115940526;
            I = w1*[w1*subs(subs(fx,s,ss1),t,tt1) + ...
                    w2*subs(subs(fx,s,ss2),t,tt1) + ...
                    w3*subs(subs(fx,s,ss3),t,tt1) + ...
                    w4*subs(subs(fx,s,ss4),t,tt1)] + ...
                w2*[w1*subs(subs(fx,s,ss1),t,tt2) + ...
                    w2*subs(subs(fx,s,ss2),t,tt2) + ...
                    w3*subs(subs(fx,s,ss3),t,tt2) + ...
                    w4*subs(subs(fx,s,ss4),t,tt2)] + ...
                w3*[w1*subs(subs(fx,s,ss1),t,tt3) + ...
                    w2*subs(subs(fx,s,ss2),t,tt3) + ...
                    w3*subs(subs(fx,s,ss3),t,tt3) + ...
                    w4*subs(subs(fx,s,ss4),t,tt3)] + ...
                w4*[w1*subs(subs(fx,s,ss1),t,tt4) + ...
                    w2*subs(subs(fx,s,ss2),t,tt4) + ...
                    w3*subs(subs(fx,s,ss3),t,tt4) + ...
                    w4*subs(subs(fx,s,ss4),t,tt4)];
            I = double(I);
        else
        end

    end
end
end
end
```

## 7.7 Appendix E: Selective Integration Method

### *7.7.1 Cantilever and Simple beams*

Both the cantilever and simple beams are done as their corresponding in section 7.4 except the line of assembling the stiffness matrix is changed to this one:

```
[element,K,centroid] =
GaussRemedyIIBilinearQuadElementStiffnessAssembly(E,NU,t,Lx,Ly,dx,dy,1
,dy/dx);
```

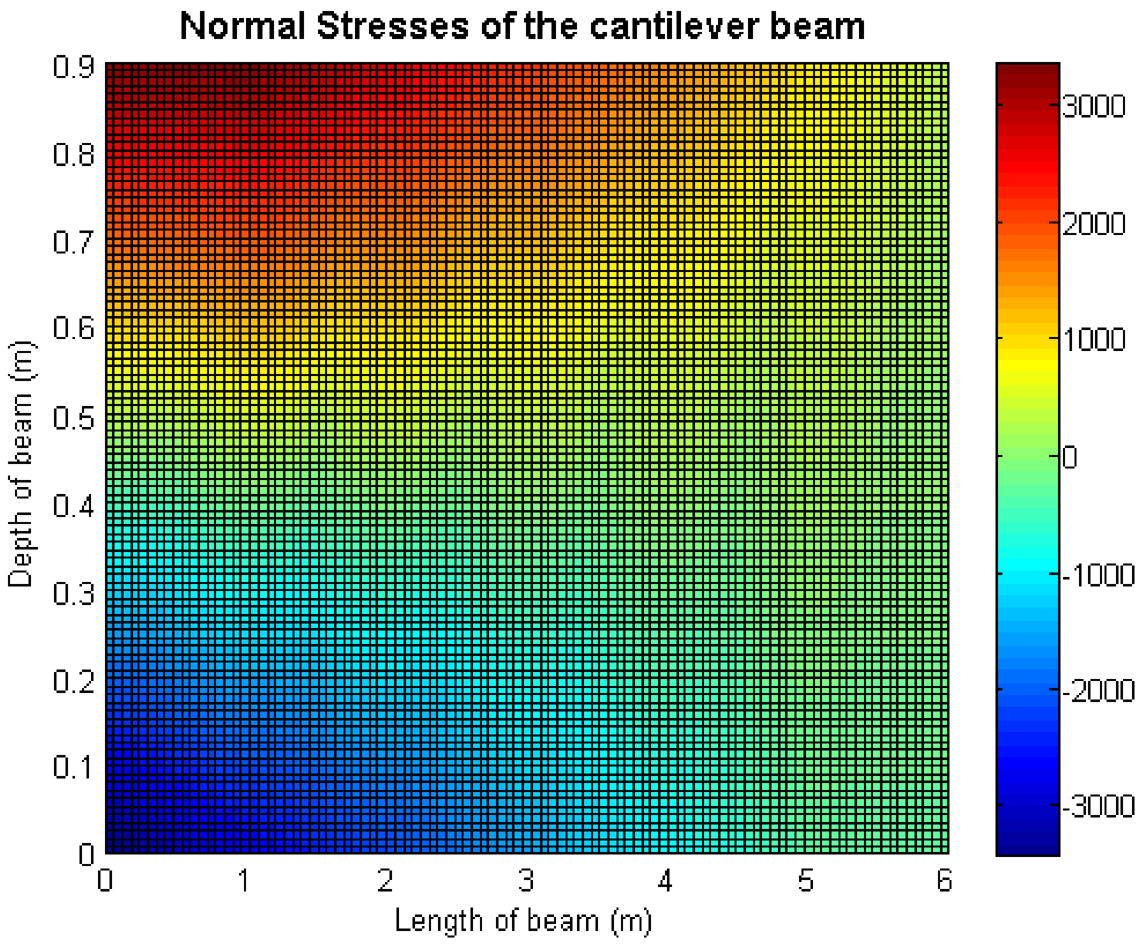


**Fig. 7-25** Analysis of CantileverBeam_C2_01_03_SelectiveInt.

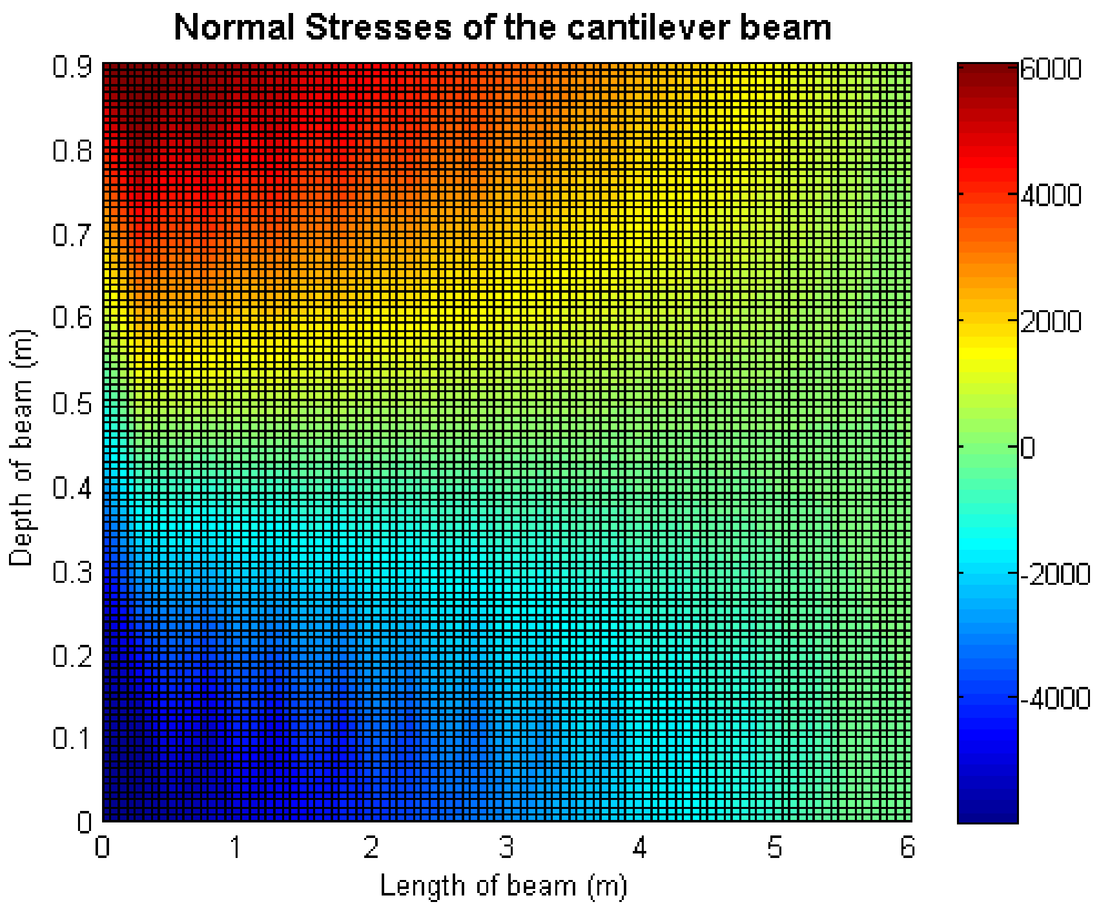


**Fig. 7-26** Analysis of CantileverBeam_C2_02_03_SelectiveInt.

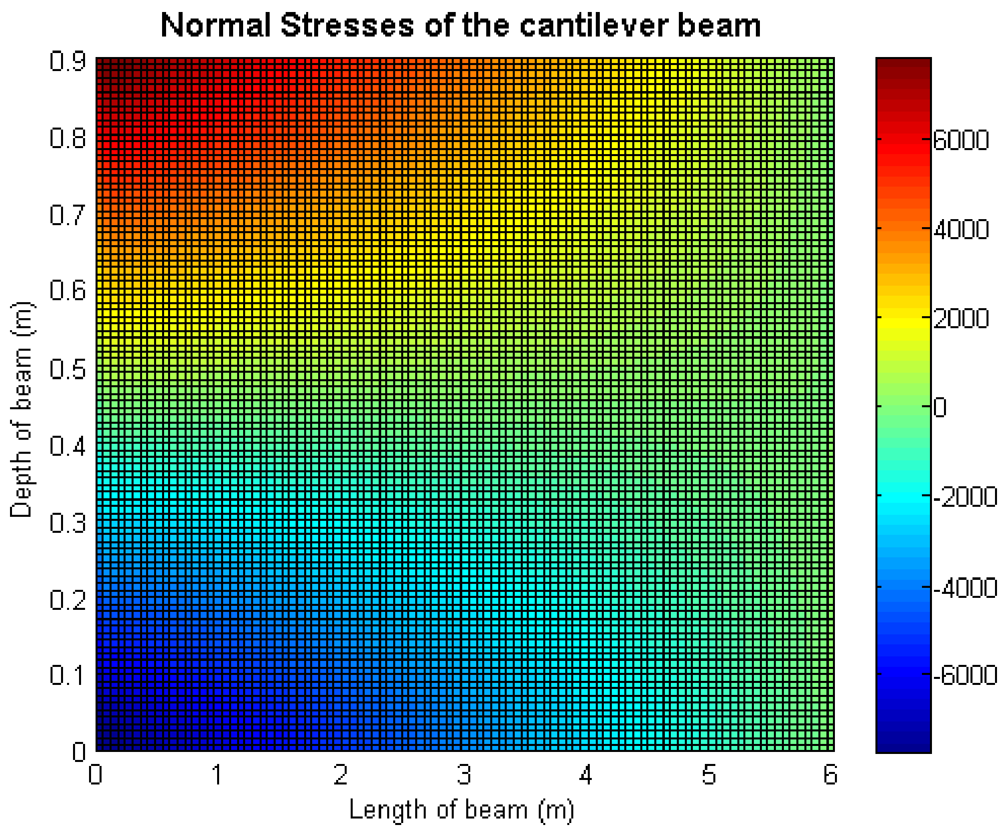
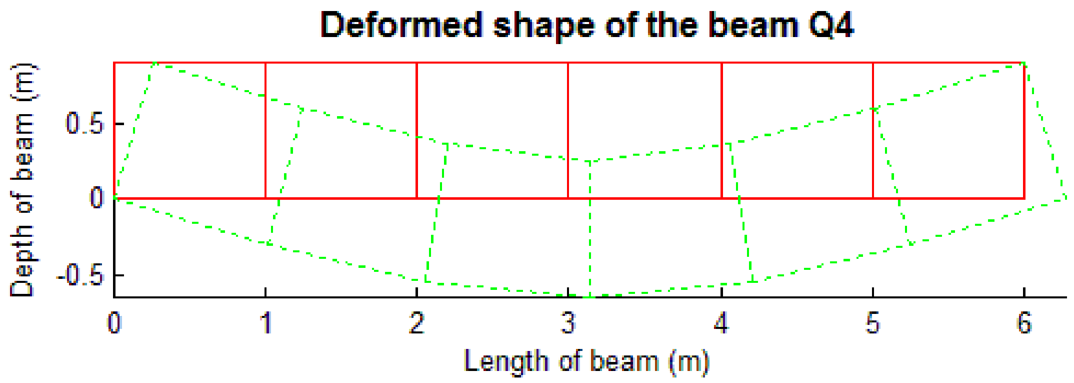
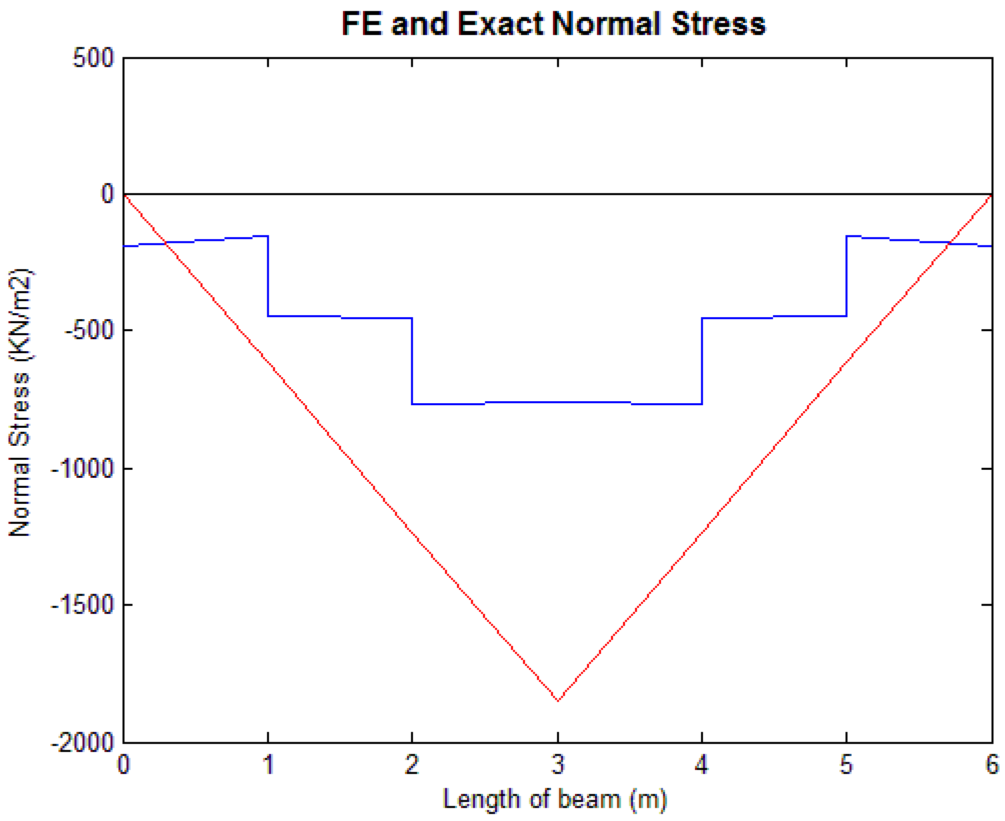


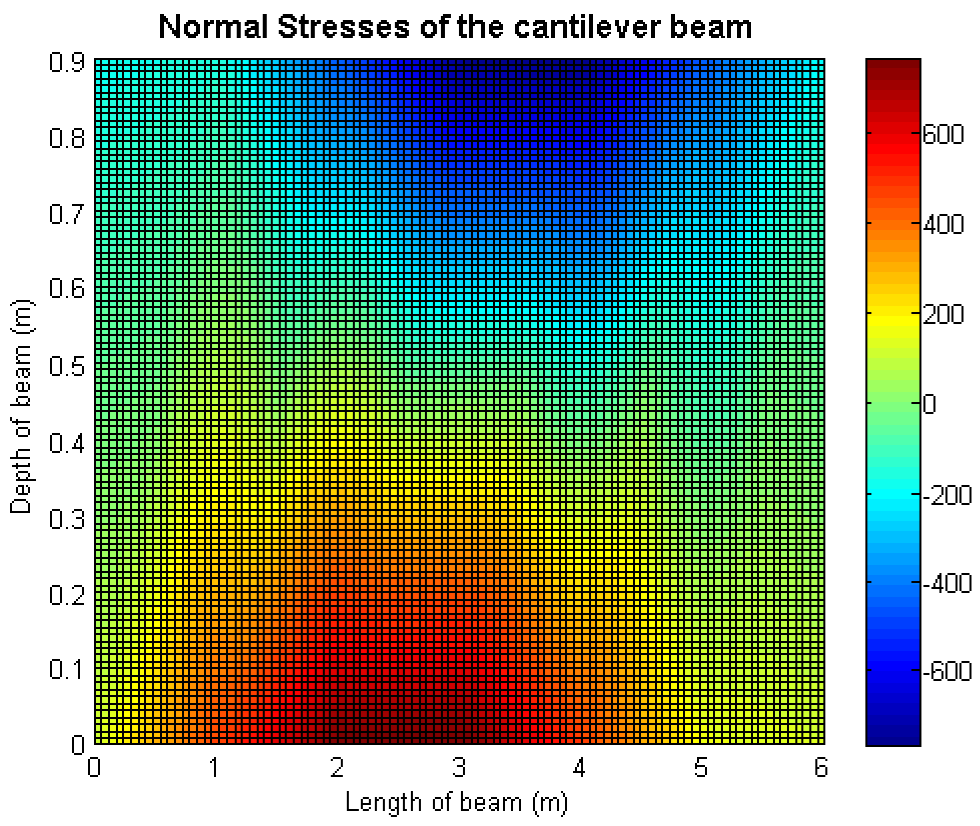
**Fig. 7-27** Analysis ofCantileverBeam_C2_03_03_SelectiveInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

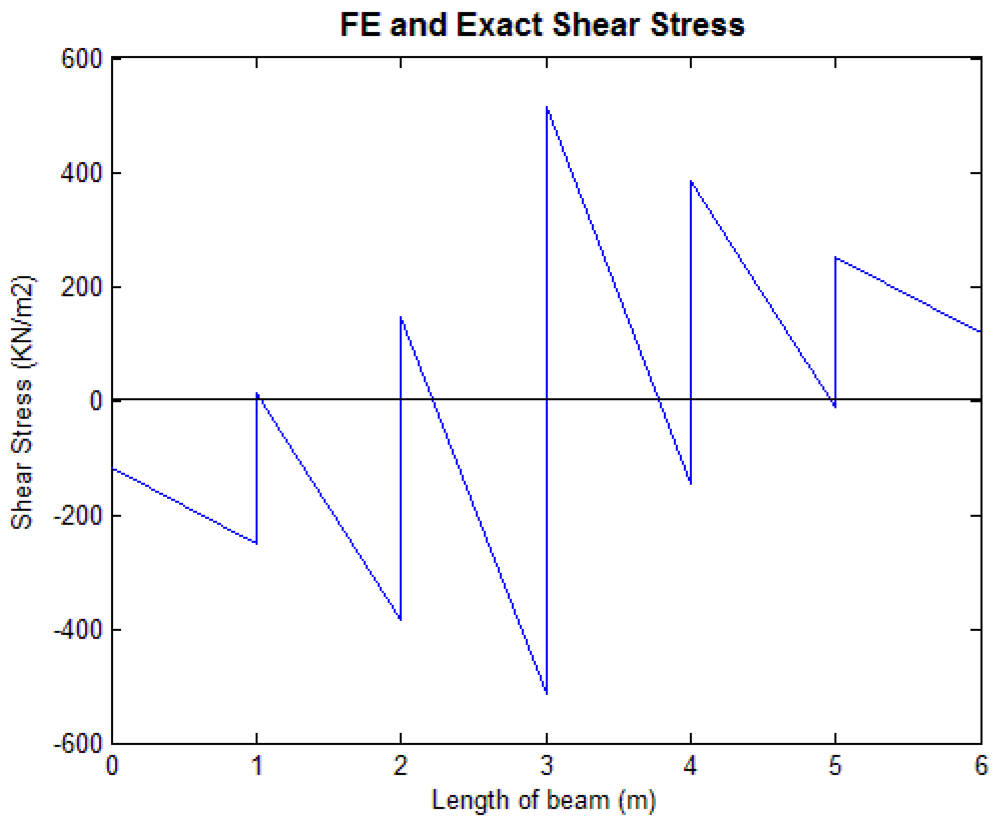
(a)


FE and Exact Normal Stress
500
0
-500
-1000
-1500
-2000
Normal Stress (KN/m2)
0
1
2
3
4
5
6
Length of beam (m)

(b)

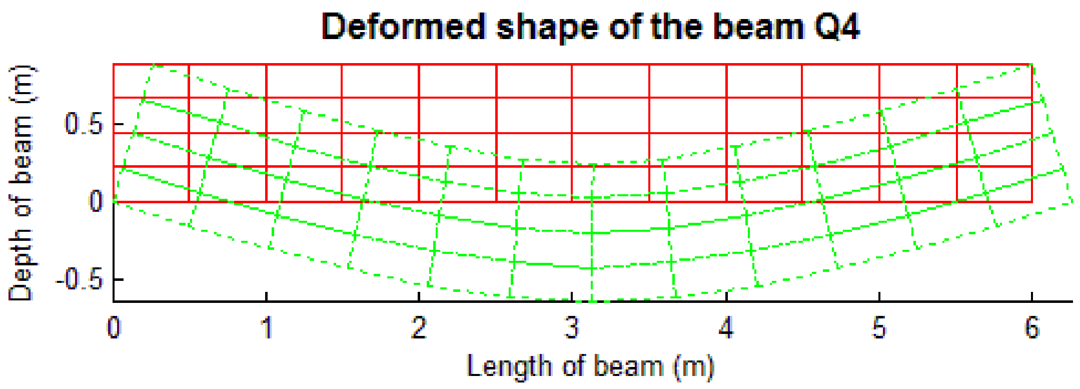


(c)

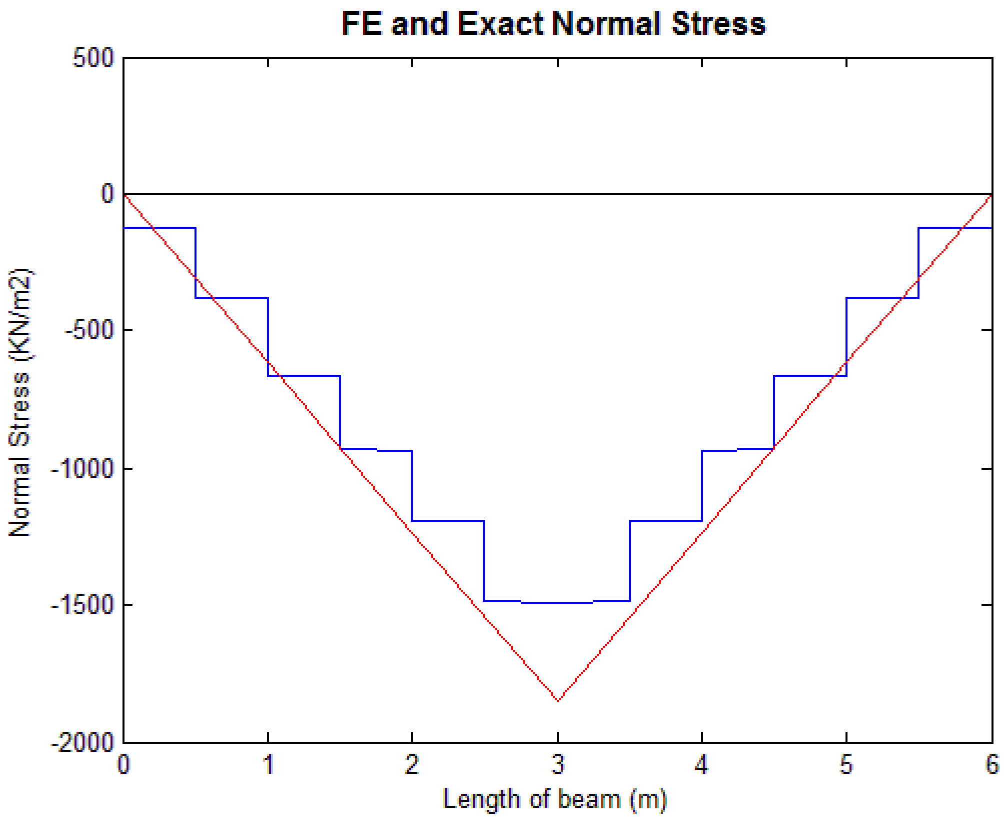


(d)

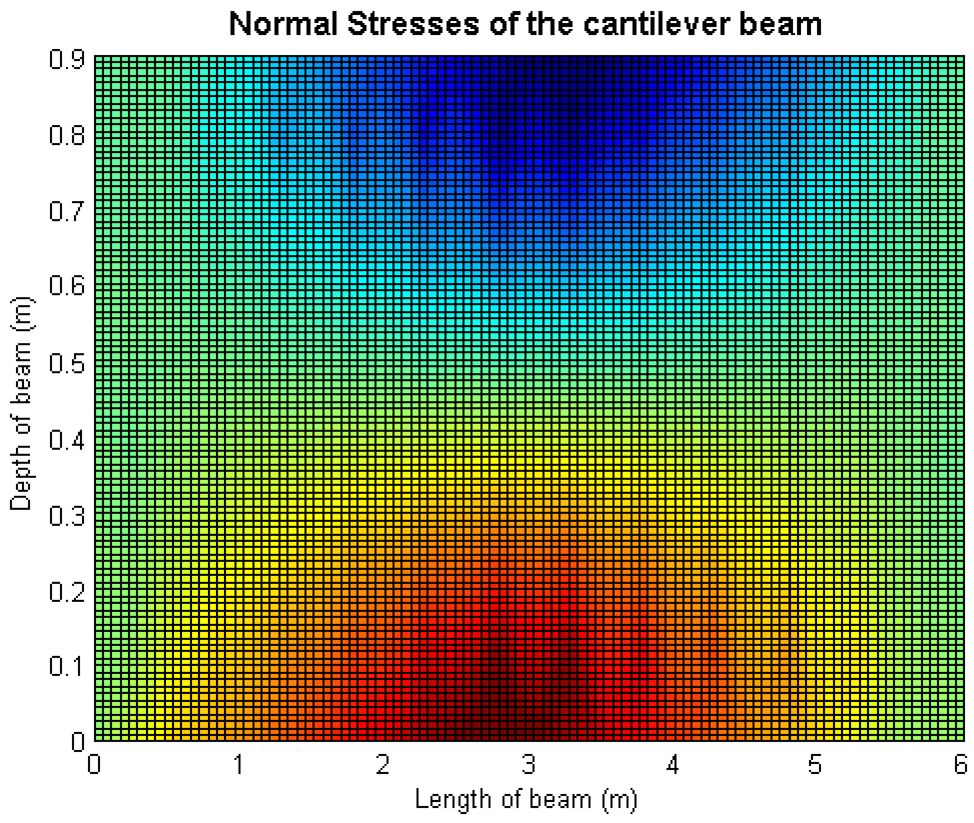
**Fig. 7-28** Analysis of SimpleBeam_A2_01_03_WeightedInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
500
0
-500
-1000
-1500
-2000
0
1
2
3
4
5
6
Length of beam (m)

(b)



(c)

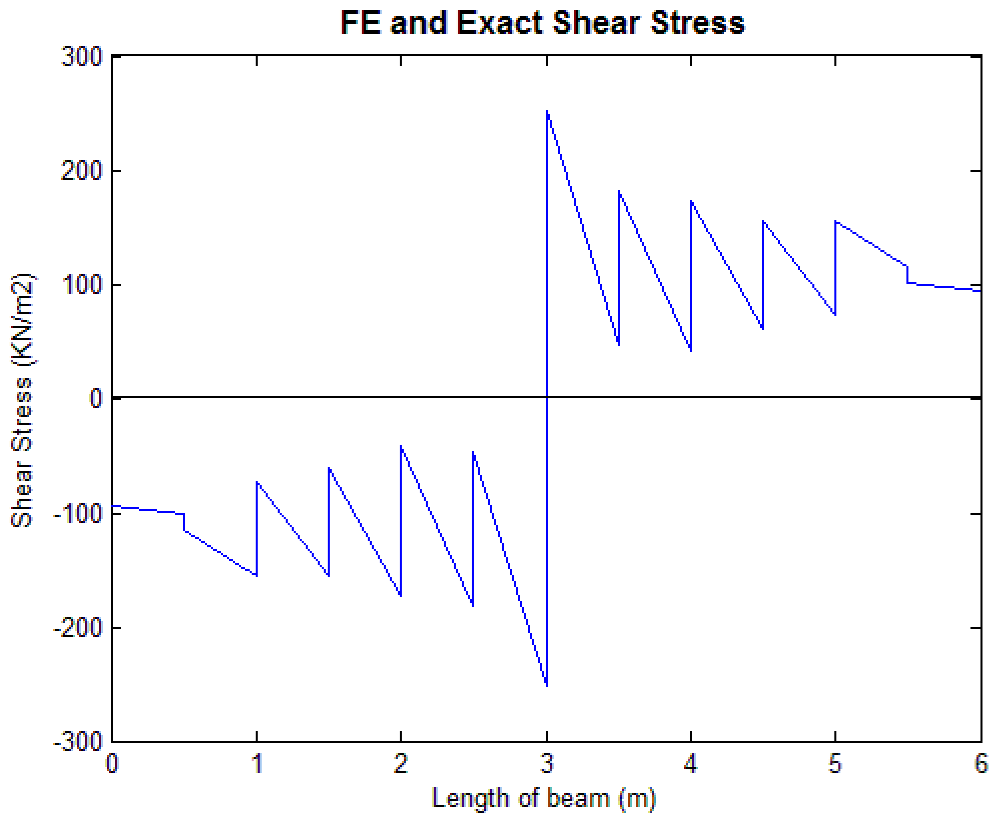
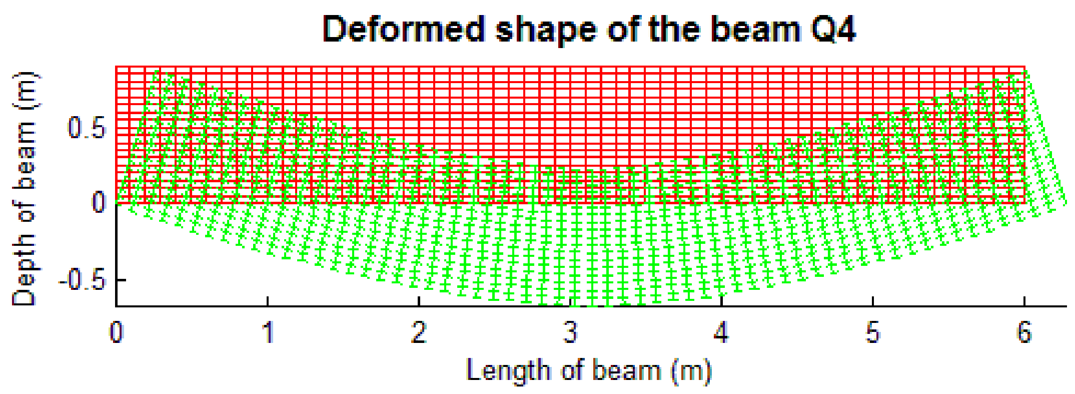
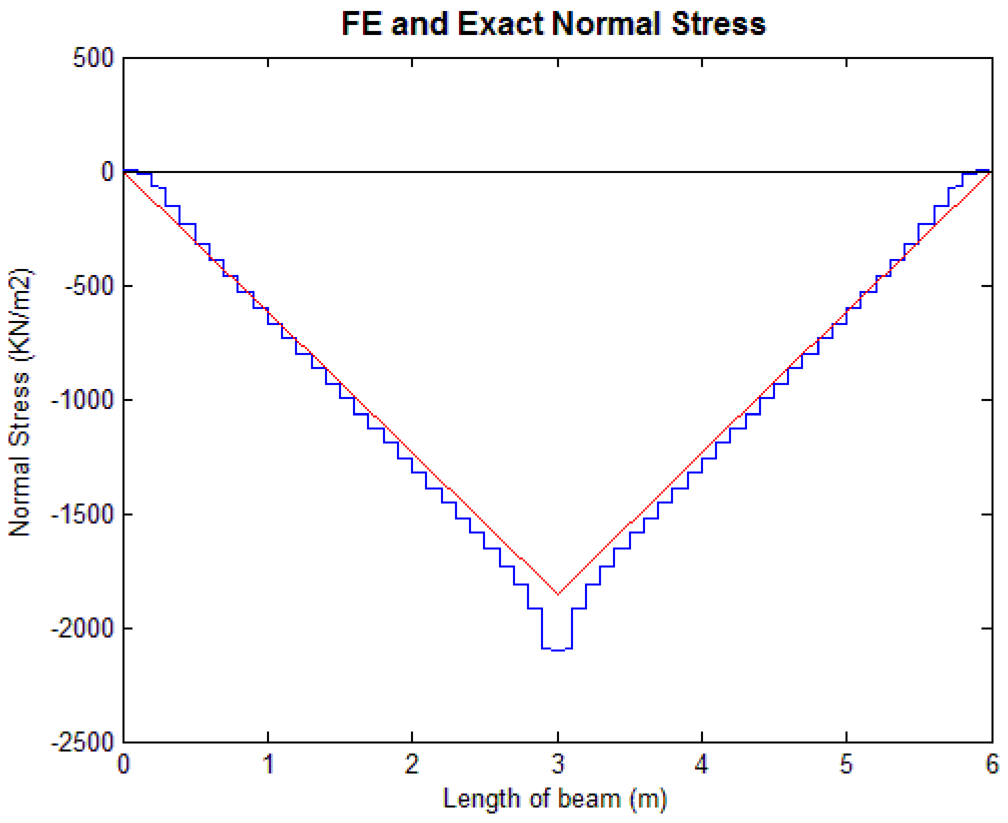


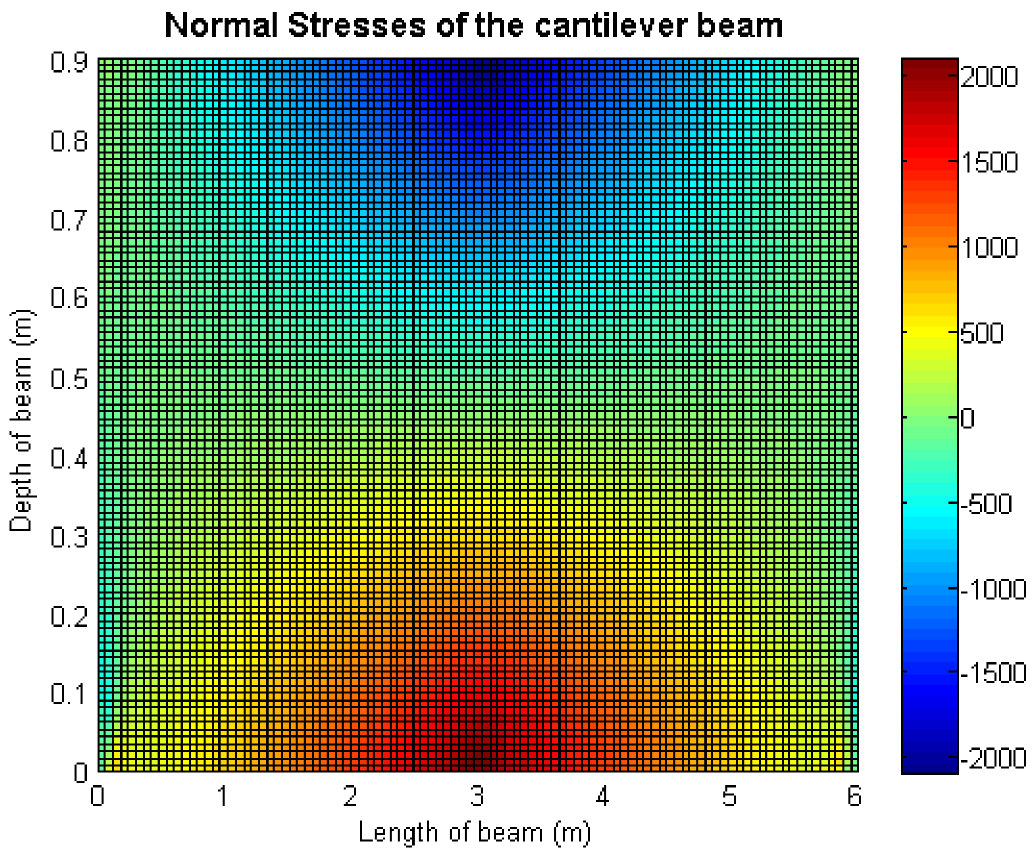
**Fig. 7-29** Analysis of SimpleBeam_A2_02_03_SelectiveInt.


Deformed shape of the beam Q4
Depth of beam (m)
0.5
0
-0.5
0
1
2
3
4
5
6
Length of beam (m)

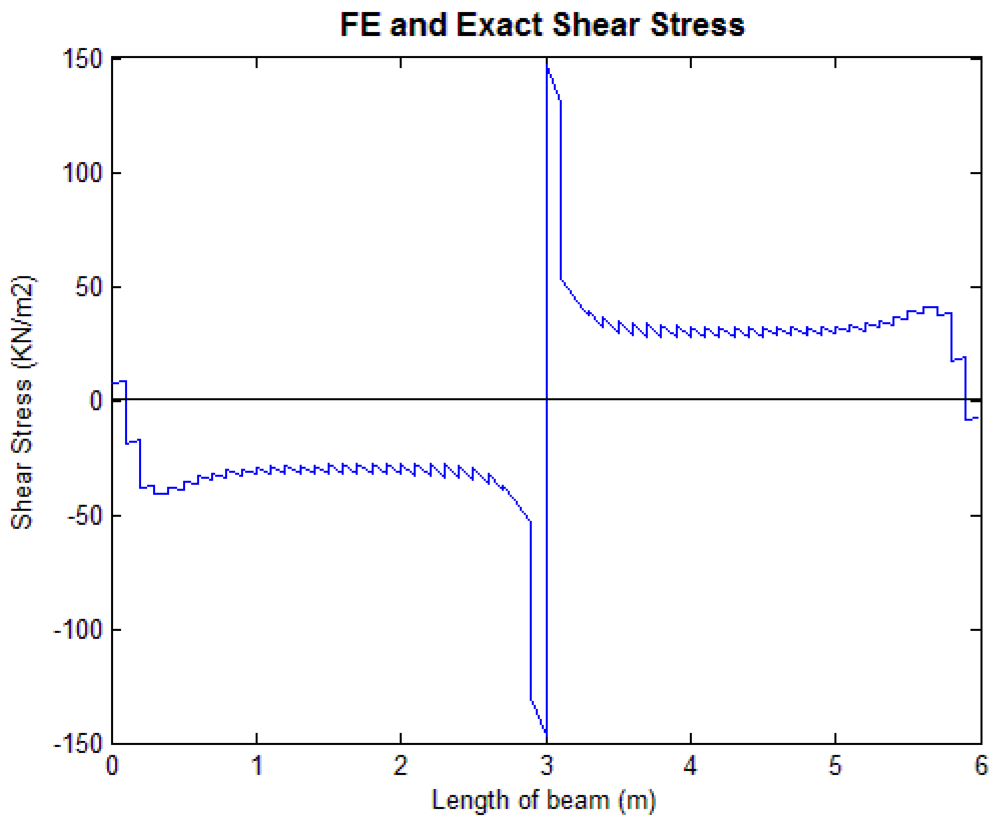
(a)


FE and Exact Normal Stress
Normal Stress (KN/m2)
500
0
-500
-1000
-1500
-2000
-2500
0
1
2
3
4
5
6
Length of beam (m)

(b)



(c)



(d)

**Fig. 7-30** Analysis of SimpleBeam_A2_03_03_SelectiveInt

### 7.7.2 Other codes

#### 7.7.2.1 Gauss Remedy II Bilinear Quadratic Element Stiffness Assembly

```
function [element,K,centroid] =
GaussRemedyIIBilinearQuadElementStiffnessAssembly(E,NU,h,Lx,Ly,dx,dy,p
,gamma)
%GaussRemedyQuadElementStiffnessAssembly This code returns the Global
%stiffness matrix for a beam/plate to be analyzed as 4-noded element
%Selective integration method.
%   The inputs are modulus of elasticity E, Poisson's ratio NU,
%thickness of the element t, length of the beam Lx, its depth Ly,
%length of element in x dx, length of the element in y Ly, p which is
%1 in case of plain stress problems and 2 in case of plain strain
%problems and gamma which is the aspect ratio. The output is the
%element matrix which contains all the details of each element for the
%beam, the whole global stiffness matrix for the whole beam K and the
%a matrix which contains the positions of the centroid of the elements
%centroid.
[element,nx,ny] = BeamMeshingQ4(Lx,Ly,dx,dy);

no = 0;
K = zeros(2*(nx+1)*(ny+1),2*(nx+1)*(ny+1));
for ii=1:size(element,1)
    i = element(ii,2);
    j = element(ii,3);
    m = element(ii,4);
    n = element(ii,5);
    x1 = element(ii,6);
    y1 = element(ii,7);
    x2 = element(ii,8);
    y2 = element(ii,9);
    x3 = element(ii,10);
    y3 = element(ii,11);
    x4 = element(ii,12);
    y4 = element(ii,13);
    centroid(ii,:) = [(x2+x1)/2,(y3+y2)/2];
    k =
GaussRemedyIIBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,
y4,p,gamma);
    mat((8*no)+1:(8*no)+8,1:8) = k;
    no = no + 1;
    K = BilinearQuadAssemble(K,k,i,j,m,n);
end
```

#### 7.7.2.2 Gauss Remedy II Bilinear Quadratic Element Stiffness

```
function k =
GaussRemedyIIBilinearQuadElementStiffness(E,NU,h,x1,y1,x2,y2,x3,y3,x4,
y4,p,gamma)
%GaussRemedyIIBilinearQuadElementStiffness This function returns the
%element stiffness matrix for a bilinear quadrilateral element with
%modulus of elasticity E, Poisson’s ratio NU, thickness h, coordinates
%of node 1 (x1,y1), coordinates node 3 (x3,y3), coordinates of node 4
%(x4,y4) and gamma which is the aspect ratio. Use p = 1 for cases of
%plane stress, and p = 2 for cases of plane strain. The stiffness
%matrix is done by Selective Integration method.
% The size of the element stiffness matrix is 8 x 8.
% Selective Integration is done by forming the stiffness matrix Ke as
%the sum of two or more matrices computed with different integration
%rules. It is more effective than Weighted Integration Method. The
%Elasticity matrix is divided into two components as what is called
%stress strain splitting where a and B are scalars; (a=v^2) and ? is
%the factor to adjust shear locking.
syms s t;
a = (y1*(s-1)+y2*(-1-s)+y3*(1+s)+y4*(1-s))/4;
b = (y1*(t-1)+y2*(1-t)+y3*(1+t)+y4*(-1-t))/4;
c = (x1*(t-1)+x2*(1-t)+x3*(1+t)+x4*(-1-t))/4;
d = (x1*(s-1)+x2*(-1-s)+x3*(1+s)+x4*(1-s))/4;
B1 = [a*(t-1)/4-b*(s-1)/4 0 ; 0 c*(s-1)/4-d*(t-1)/4
    c*(s-1)/4-d*(t-1)/4 a*(t-1)/4-b*(s-1)/4];
B2 = [a*(1-t)/4-b*(-1-s)/4 0 ; 0 c*(-1-s)/4-d*(1-t)/4
    c*(-1-s)/4-d*(1-t)/4 a*(1-t)/4-b*(-1-s)/4];
B3 = [a*(t+1)/4-b*(s+1)/4 0 ; 0 c*(s+1)/4-d*(t+1)/4
    c*(s+1)/4-d*(t+1)/4 a*(t+1)/4-b*(s+1)/4];
B4 = [a*(-1-t)/4-b*(1-s)/4 0 ; 0 c*(1-s)/4-d*(-1-t)/4
    c*(1-s)/4-d*(-1-t)/4 a*(-1-t)/4-b*(1-s)/4];
Bfirst = [B1 B2 B3 B4];
Jfirst = [0 1-t t-s s-1 ; t-1 0 s+1 -s-t ;
    s-t -s-1 0 t+1 ; 1-s s+t -t-1 0];
J = [x1 x2 x3 x4]*Jfirst*[y1 ; y2 ; y3 ; y4]/8;
B = Bfirst/J;
alpha = NU^2;
%beta is a factor to adjust shear locking
beta = ((2/gamma^2)*(1-NU^2))/(1+(2/(gamma^2))-NU);
if p == 1
    DI = (E/(1-NU*NU))*[alpha, beta, 0; beta, alpha, 0; 0, 0, (1-
NU)/2];
    DII = (E/(1-NU*NU))*[1-alpha, NU-beta, 0; NU-beta, 1-alpha, 0; 0,
0, 0];
elseif p == 2
    D = (E/(1+NU)/(1-2*NU))*[1-NU, NU, 0; NU, 1-NU, 0;
    0, 0, (1-2*NU)/2];
end
BDI = J*transpose(B)*DI*B;
BDII = J*transpose(B)*DII*B;
%Integration of stiffness matrix using 1 gauss point
k1x1 = intGQ(1,BDI);
%Integration of stiffness matrix using 2 gauss points
k2x2 = intGQ(2,BDII);
%Stiffness matrix by weighted integration method
kB = (k1x1)+(k2x2);
z = h*kB;
k = double(z);
end
```

# CHAPTER 8

## BIBLIOGRAPHY

# 8. Bibliography

[1] C. A. Felippa, "Introduction to Finite Element Methods," 2004. [Online]. Available: http://www.colorado.edu/engineering/cas/courses.d/IFEM.d/IFEM.Ch00.d/IFEM.Ch00.pdf. [Accessed 13 April 2014].

[2] R. D. Cook, Finite Element Modeling For Stress Analysis, 1st ed., Madison: John Wiley & Sons, Inc., 1994, p. 1.

[3] A. E. Armenakas, Advanced Mechanics of Materials And Applied Elasticity, CRC Press, 2005.

[4] D. Braess, Finite elements: theory fast solvers and applications in solid mechanics, 2nd ed., Bochum: Springer Verlag, 1992.

[5] D. S. M. M. E. P. R. J. W. Robert D. Cook, Concepts and applications of finite element analysis, 4th ed., Madison: John Wiley & Sons. Inc., 2002.

[6] U. o. N. Department of Engineering Mechanics, "Pure Bending," 1996. [Online]. Available: http://emweb.unl.edu/NEGAHBAN/Em325/11-Bending/Bending.htm. [Accessed 30 May 2014].

[7] R. L. T. W. P. D. a. J. G. E. L. Wilson, "Incompitable Displacement Methods," *Numerical and Computer Methods in Structural Mechanics,* pp. 43-57, 1973.

[8] E. L. Wilson, "Ed Wilson Website," [Online]. Available: http://www.edwilson.org/BOOK-Wilson/06-incom.pdf. [Accessed 17 June 2014].

[9] P. I. Kattan, MATALB Guide to Finite Elements, 2nd ed., Amman: Springer, 2006.

[10] B. McGinty, "Finite Deformation Continuum Mechanics," 2012. [Online]. Available: http://www.continuummechanics.org/cm/hydrodeviatoricstress.html. [Accessed 30 June 2014].

[11] B. Vossen, "Volumetric locking in finite elements," Eindhoven, 2008.

[12] P. Schreurs, Applied Elasticity in Engineering, Eindhoven, 2013.

[13] L.-D. Support, "A pathological case of volume locking in triangular elements," [Online]. Available: http://www.dynasupport.com/tutorial/element-locking/a-pathological-case-of-volume-locking-in-triangular-elements. [Accessed 3 June 2014].

[14] T. J. R. Hughes, The Finite Element Method, 2nd ed., Stanford: Dover Publications, 2000.

[15] A. Ferreira, MATLAB Codes for Finite Element Analysis, Porto: Springer Science, 2008.

[16] K.-J. Bathe, Finite Element Procedures, New Jersy: Prentice Hall, 1982.

[17] M. Papadrakakis, "Analysis of Structures of Finite Element Method," Athens, 2012.