AUGMENTED CLOTH FITTING WITH REAL TIME PHYSICS SIMULATION USING TIME-OF-FLIGHT CAMERAS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING
AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BİLKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

By
Umut Gültepe
December, 2012

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

(Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

AUGMENTED CLOTH FITTING WITH REAL TIME PHYSICS SIMULATION USING TIME-OF-FLIGHT CAMERAS

Umut Gültepe

M.S. in Computer Engineering

Supervisors: Assoc. Prof. Dr. Uğur Güdükbay

December, 2012

This study surveys and proposes a method for an augmented cloth fitting with real time physics simulation. Augmented reality is an evolving field in computer science, finding many uses in entertainment and advertising. With the advances in cloth simulation and time-of-flight cameras, augmented cloth fitting in real-time is developed , to be used in textile industry in both design and sale stages. Delay in cloth fitting due to processing time is the main challenge in this research. Human body is identified, articulated and tracked with a time-of-flight camera. Depending on the size and position of body limbs, a virtually simulated cloth is fitted in real time on the subject. Delay is reduced with GPU computing for cloth simulation and collision detection.

Keywords: cloth simluation, computer vision, natural interaction, virtual fitting room, kinect, depth sensor.

ÖZET

UÇUŞ ZAMANI KAMERALARI KULLANAN GERÇEK ZAMANLI FİZİK SİMÜLASYONLU ARTIRILMIŞ KIYAFET KABİNİ

Umut Gültepe
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticileri: Doç. Dr. Uğur Güdükbay

Aralık, 2012

Bu çalışma fizik simülasyonlu bir artırılmış kıyafet giydirme için bir metot önermekte ve incelemektedir. Artırılmış gerçeklik bilgisayar biliminde gelişen bir alandır, eğlence ve reklam sektörlerinde geniş yer bulmaktadırlar. Kumaş simülasyonu ve uçuş-zamanı kameralarının geliştirilmesi ile, gerçek zamanlı artırılmış kıyafet giydirimi tekstil endüstrisinde tasarım ve satış aşamalarında kullanılmak üzere geliştirilmiştir. şleme zamanı sebebiyle kıyafet giydirilmesindeki gecikme bu araştırmadaki en büyük zorluktur. nsan vücudu bir uçuş-zamanı kamerası ile tanımlanmakta, bölünmekte ve takip edilmektedir. Vücut parçalarının boyutlarına ve pozisyonlarına göre simüle edilen bir sanal kıyafet kullanıcının üstüne yerleştirilmektedir. Gecikme zamanı kıyafet simülasyonunu ve çarpışma takibini GPU üzerinde yaparak azaltılmaktadır.

Anahtar sözcükler: kıyafet simülasyonu, bilgisayarla görü, doğal etkileşim, sanal giyinme kabini, kinect, derinlik sensörü.

Acknowledgement

I am deeply grateful to my supervisor Assoc. Prof. Dr. Uğur Güdükbay, who guided and assisted me with his invaluable suggestions in all stages of this study. I also chose this area of study by inspiring from his deep knowledge over this subject.

I am very grateful to Computer Engineering Department of Bilkent University for providing me scholarship for my MS study.

I would like to thank Scientific and Technical Research Council of Turkey (TÜBİTAK) and the Turkish Ministry of Industry and Technology for their financial support for this study and MS thesis.

to my mother, father and my brother...

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Overall Framework-OGRE

As a programmer, I embrace the do not repeat yourself mindset and extreme programming methodology. Furthermore, since my study required utilization of many different techniques in different fields in Computer Science, I searched for various 3rd party SDKs to save me from writing ground-level code. These two beliefs led me to search for the rendering engine best suited for my needs:

1. Must be code oriented rather than designer oriented.
2. Must be able to utilize both DirectX and OpenGL (for compatibility reasons).
3. Must take care of mundane and routine programming such as the rendering pipeline, input handling,
4. Must be able to integrate easily with 3rd party libraries.
5. Must be stable and mature.
6. Must have an associated 3-D designing program which can be used to easily produce content and load into the program.
7. Must have accurate and extensive documentation.

Other than these, automatic material rendering, skeletal animation support were considered as bonuses. After experimenting with Unity, UDK and native OpenGL programming, I eventually decided on Object-Oriented Rendering Engine (will be referred as OGRE in the rest of the paper) for it complies with my requirements the best.

1.1 The Features

Unlike the name suggests, OGRE is more than just a rendering engine. Among all the features, the ones I utilized are as following [4]

- Render State Management
- Spatial Culling and Transparency Handling
- Material Rendering
  - Easy material and shader management, custom shader support.
  - Multitexture and Multipass blending
  - Lighting shader and different shadow rendering techniques
  - Material Level Of Detail Support
  - Supports a variety of image formats, volumetric textures and DXT textures.
  - Render-To-Texture (Frame Rendering Buffer) support
- Meshes
  - Native mesh format which can be exported from Blender Designer.
  - Level Of Detail Support
  - Skeletal animation feature, can be used with models exported from Blender.
    - Multiple-bone weighted skinning

    * Hardware Acceleration.

- Easy scene management
- Easy camera and input management.
- Easy integration with 3rd party libraries due to code-based nature.
- Overlay feature which enables easy information tracking about the feature.

These features led me to choose OGRE as my base framework.

1.2 High Level Overview

The class diagram in Figure 1.1 shows the Object-Oriented core of the OGRE [5]:

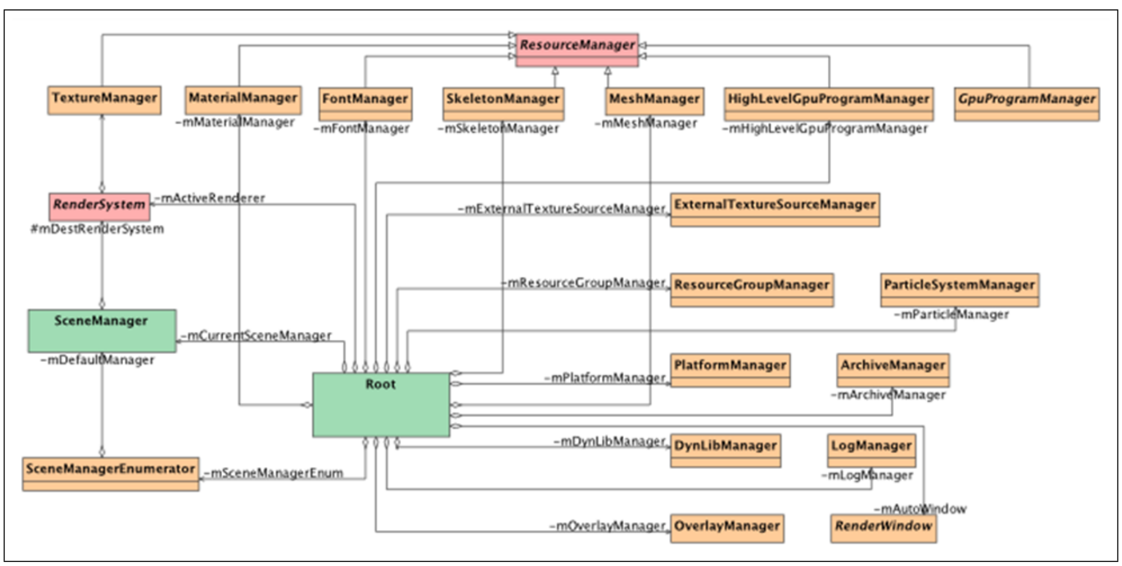

Figure 1.1: OGRE High Level Overview

1.2.1 The Root object

The root object is the entry point and core of the framework.

- It is created first and destroyed last in the application lifecycle.
- It configures the system, delivers pointers to the managers for various resources.
- Provides automatic rendering cycle, continued until an interrupt from FrameListener objects.

1.2.2 The RenderSystem Object

Render system is an abstract class to define the underlying 3D API (either Direct 3D or OpenGL). This class is not accessed and modified by the application programmer.

1.2.3 The SceneManager Object

SceneManager is the most used object by the application programmer, as it is in charge of the contents in the scene to be rendered.

- It is used to create, destroy and update the objects.
- It sends the scene to the RenderSystem behind the curtains for rendering.
- Multiple SceneManagers can be used to create other visual resources (ex. RenderToTexture environment)

1.2.4 Resource Manager

ReosurceManager object is an abstract class, used to create, keep and dispatch a type of resource it is associated with.

- The associated type is defined by the class inheriting the ResourceManager, such as MaterialManager.

- There is always only one instance of every child of ReosurceManager in an application.
- Resource Managers search the pre-defined locations of the filesystem and automatically indexes the resources available, ready to be loaded upon demand.

1.2.5 Entities, Meshes and Materials and Overlays

Entities are the instances of movable objects in the scene. They are based on meshes, which define the geometric and material properties of the entity. Materials contain information about what color the final pixel in the rendering should be.

- Entities are attached to scene nodes for moving and rendering. Scene nodes can be nested, which greatly simplifies the process of rendering complex scenes.
- Meshes consist of submeshes, which can have different material associations. Therefore, a mesh can be composed of various parts with various materials.
- Materials are defined either in run-time or in .material scripts, with detailed information. They also support custom shaders.
- Mesh files can be created and saved with manual objects, or exported through designer programs such as Blender. The .mesh files are in binary format.

Overlays are used to create panels for control and HUD which are rendered above the scene. They are 2D elements are placed either by screen proportion or pixel size and rendered orthographically, last in the rendering pipeline by default (this can be overridden).

1.2.6 The render cycle

The self-explanatory render cycle of OGRE is given in Figure 1.2 [5].

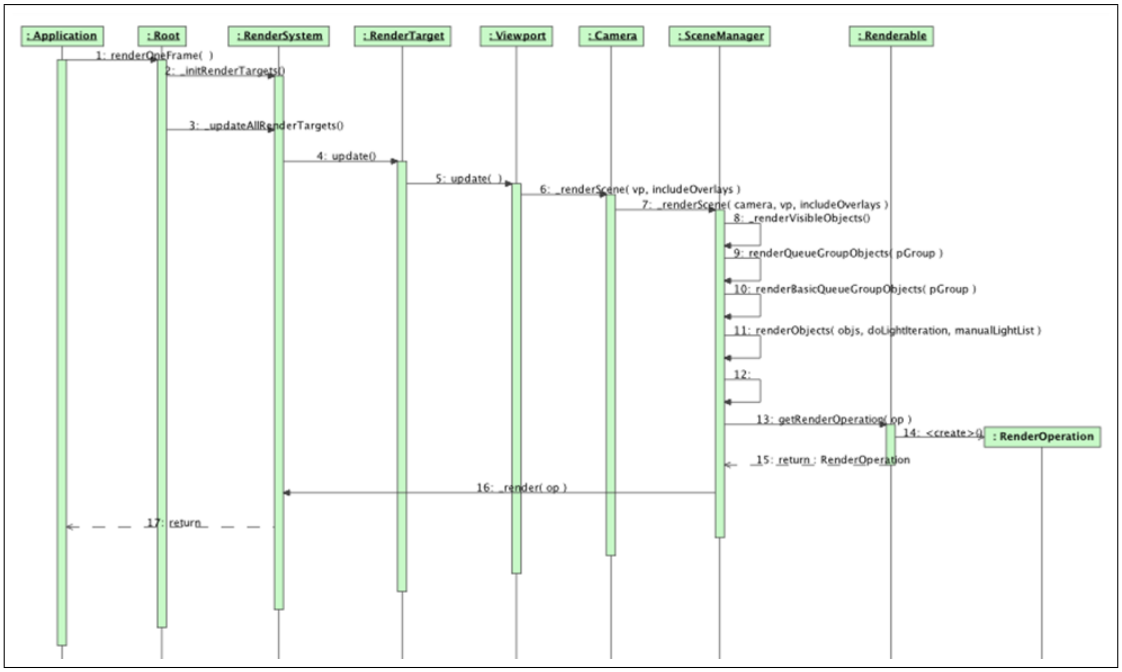

Figure 1.2: OGRE Render Cycle

Chapter 2

User Tracking

2.1 Mesh Deformation Techniques

User tracking has always been both a challenge and a valued feature in Image Processing. Until the availability of Time-Of-Flight cameras, user tracking was dependent on RGB cameras. Although RGB cameras are sufficient for user tracking, they are proven to be harder to use for body articulation and joint estimation, mostly because of the complexity of human body, self occlusions and the difficulties of body segmentation based on pixel colors alone. Researchers started with still images, than experimented with image sequences from multiple cameras. Most common technique was to extract the body silhouette from the image and construct a body shape with silhouettes from multiple calibrated cameras-Shape From Sihouette (SFS) [2]. SFS algorithms evolved from spatial to temporal tracking and their accuracy improved even more [2].

However, RGB based accurate user articulation techniques required many cameras-a financial problem. After the need for extensive imaging hardware, the processes required to calibrate the cameras initially, extract and combine the silhouettes, build a shape and articulate the result. These processes require very complex algorithms, many man-hours to implement and very powerful computing infrastructure. All these requirements made shape-from-silhouette algorithms hard to utilize for me.

Instead of RGB imaging, I searched for an alternate type of device which can capture the depth of the field in front of it. Although there are a variety of such devices, from sonars to Laser Scanners, the one most appropriate for user tracking was Time-Of-Flight cameras. They have significant advantages over Stereo-Vision and Laser Tracking in simplicity, are fast (up to 100 fps) and accurate in distance [3].

Choosing the most accurate Time-Of-Flight camera was easy, as Microsoft Kinect was not only the cheapest and the most available of them all, it was also the most powerful and had an extensive developer community. I utilized the Kinect for XBOX rather than Kinect for Windows in this study, due to its distance and performance characteristics.

2.2 Software

The user tracking process with Kinect was much simpler compared to SFS techniques, due to the Shape being available mostly with the depth field output. However, proper articulation still required lots of different algorithms and time.

In order to speed up the user tracking and articulation, I utilized the OpenNI framework along with PrimeSense NITE package and SensorKinect drivers. OpenNI provides the framework for capturing and utilizing the various types of streams from Natural Interaction devices, also provides abstract modules for accessing complex functionalities, such as skeletal tracking [6]. PrimeSense NITE package is a software that bundles with OpenNI, and provides skeletal tracking, hand tracking and other functionalities which can be accessed via OpenNI framework [7].

With the integration of these modules to my application, I was able to acquire the joint positions and orientations from the depth sensor with almost no effort except the integration itself. With the current hardware/software configuration, I acquire 15 joint positions and orientations at 30 fps, which is enough to reproduce the movement of the user on the virtual model. Other than the joint information, I also acquire the depth and image streams from the depth camera. I will be using the image stream to present the results I have obtained, comparing the user movement with the simulated environment.

Chapter 3

Hand Tracking

3.1 OpenCV

Although OpenNI/NITE provided me the joint information, useful functions for a smooth user interface, such as hand state recognition or hand swipe filters were omitted in the open source natural interaction libraries I use. In order to simulate mouse-clicking behavior, I decided to track the hands of the user to be used as cursors, notice open/close hand gestures for clicking events. In order to find implementations of the algorithms in my proposed hand-track solution, I researched various libraries which came with functions implementing such algorithms. I chose to work with OpenCV, due to its maturity, community and integration with other libraries.

OpenCV not only provides basic functions to perform complex and processor-intensive image processing functions such as, facial recognition system, gesture recognition, stereoscopic 3D and segmentation, it also has a very vast machine learning aspect, including boosting, decision three learning and many more features [1].

3.2 The Process

Utilizing such a powerful middleware as a depth sensor, I am able to perform very robust background subtraction with almost no effort skeletal body tracking is another embedded property of the software which I use, giving me a speed boost. Therefore, I implemented two different techniques: Hand state recognition and hand swipe recognition.

Hand swipe recognition is simpler compared to hand state recognition. It is just a matter of keeping track of the hands 3D position, and keeping a listener which is activated when the 3D velocity of the hand exceeds a certain number. I also performed a number of optimizations on this function to make it invariant with the size and location of the user.

Open/Close Hand recognition is harder than Hand Swipe recognition. I had to perform advanced image processing in order to determine the state of the hand successfully, at relatively low resolutions and large distances. My algorithm is shown in Figure 3.1:

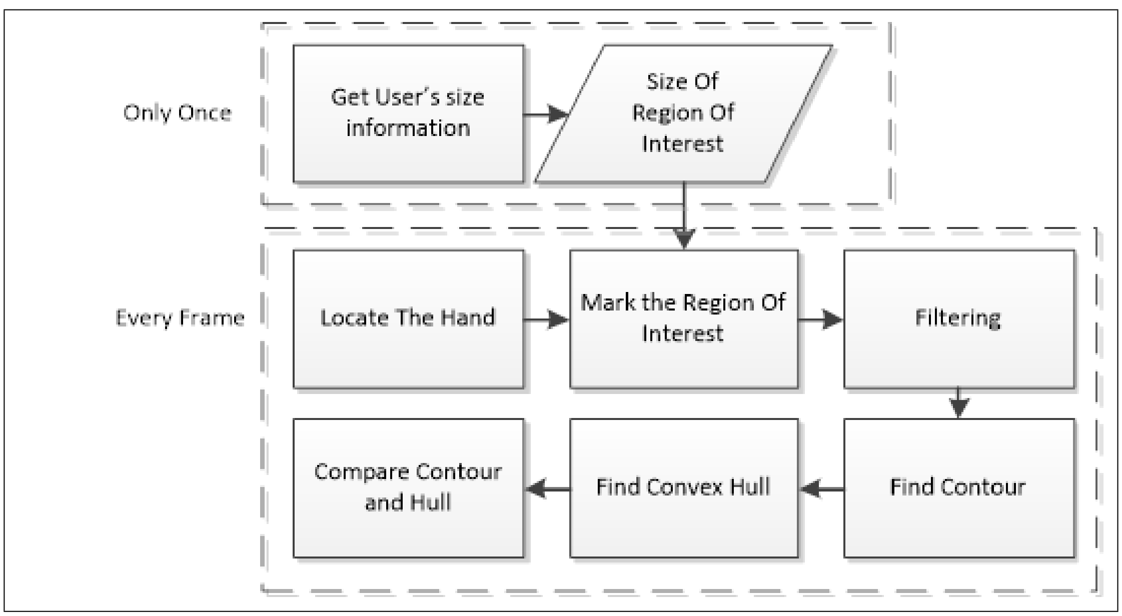

Figure 3.1: Hand Recognition Algorithm

1. The size information for the region of interest is drawn from the distance

between the users head and users neck.

2. The hand is located using skeletal body tracking.

3. The marked region is copied from the depth stream

4. Two dilation and one erosion operations are performed to smooth the hand image

5. The contour is found on the filtered image

6. The convex hull of the found contour is calculated.

7. The depth difference between the hull and the actual contour is taken as the reference for hand-state.

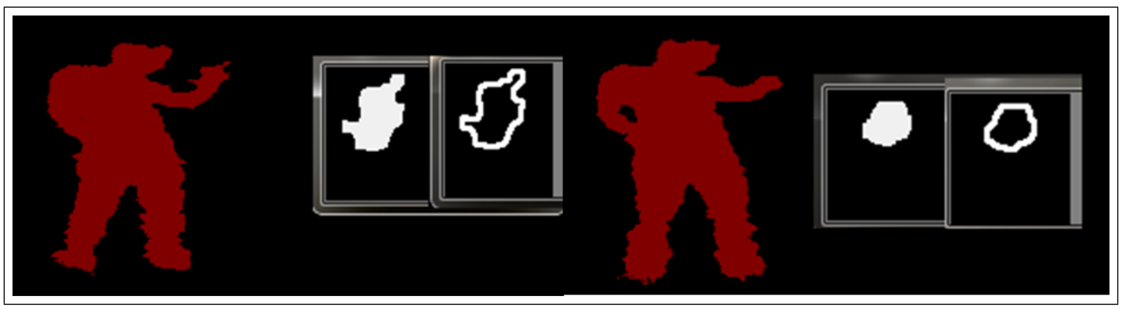

Figure 3.2: Hand Images and Contours from Depth Stream

The test run results can be seen in figure Figure 3.2. The left part of the figure shows the hand in open state, whereas the hand is closed on the right part.

Ultimately, I plan to implement the algorithm described in [8], where the author claims to have achieved 96% correct results. I have not included RGB image in the process yet, neither have I implemented complex machine learning. In addition to these methods, I will record hand images for testing objectively. These topics are in my future research plans.

Chapter 4

Linear Finite Element Method

This chapter describes in detail the linear FEM method, development of small deformation strains that lead to linear FEM, the stiffness matrix, and the solution of the linear FEM.

4.1 Linear FEM Using Tetrahedral Elements

Most of the linear FEM methods for 3D tetrahedral elements consist of five stages:

1. Tetrahedralization;
2. Construction of elemental stiffness matrices;
3. Assembly of elemental stiffness matrices and force vectors;
4. Applying boundary conditions;
5. Solving the linear system that gives unknown nodal displacements.

4.1.1 Tetrahedralization

We use same tetrahedralized meshes for both linear and non-linear solutions to provide integrity and make a better analysis for our experiments. We use the Application Programming Interface of TetGen [?] as a tetrahedral mesh generator and integrated it into our implementation. The tetrahedralization process produces the nodal positions and the elements for each node. We use nodal positions to construct elemental stiffness matrices, and element's nodal information to assemble the elemental stiffness matrices to form the global stiffness matrix.

4.1.2 Construction of Elemental Stiffness Matrices

□

Figure 4.1: Tetrahedral element

We use tetrahedral elements for modeling meshes in the experiments (Figure 4.1). Overall, there are 12 unknown nodal displacements in a tetrahedral element. They are given by [?]

$$\left\{d\right\} = \left\{u(x,y,z)\right\} = \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ \vdots \\ u_4 \\ v_4 \\ w_4 \end{Bmatrix}. \tag{4.1}$$

In global coordinates, we represent displacements by linear function by

$$u^e(x,y,z) = c_1 + c_2 x + c_3 y + c_4 z. \tag{4.2}$$

For all 4 vertices, Equation 4.2 is extended as

$$\begin{bmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{bmatrix} \begin{Bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{Bmatrix} = \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}. \tag{4.3}$$

Constants c_n can be found as

$$c_n = v^{-1} u_n, \tag{4.4}$$

where v^{-1} is given by

$$v^{-1} = \frac{1}{det(v)} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix}. \tag{4.5}$$

$det(v)$ is $6V$, where V is the volume of the tetrahedron. If we substitute Equation 4.4 into Equation 4.2, we obtain

$$u^e(x, y, z) = \frac{1}{6V^e} \begin{bmatrix} 1 & x & y & z \end{bmatrix} \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\ \beta_1 & \beta_2 & \beta_3 & \beta_4 \\ \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\ \delta_1 & \delta_2 & \delta_3 & \delta_4 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}. \tag{4.6}$$

α, β, γ, δ and the volume V are calculated by

$$\alpha_1 = \begin{vmatrix} x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}, \quad \alpha_2 = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{vmatrix}, \quad \alpha_3 = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_4 & y_4 & z_4 \end{vmatrix}, \quad \alpha_4 = -\begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} \tag{4.7}$$

$$\beta_1 = -\begin{vmatrix} 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix}, \quad \beta_2 = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_3 & z_3 \\ 1 & y_4 & z_4 \end{vmatrix}, \quad \beta_3 = -\begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_4 & z_4 \end{vmatrix}, \quad \beta_4 = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \tag{4.8}$$

$$\gamma_1 = \begin{vmatrix} 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{vmatrix}, \quad \gamma_2 = -\begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_3 & z_3 \\ 1 & x_4 & z_4 \end{vmatrix}, \quad \gamma_3 = \begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_4 & z_4 \end{vmatrix}, \quad \gamma_4 = -\begin{vmatrix} 1 & x_1 & z_1 \\ 1 & x_2 & z_2 \\ 1 & x_3 & z_3 \end{vmatrix} \tag{4.9}$$

$$\delta_1 = -\begin{vmatrix} 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{vmatrix}, \quad \delta_2 = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \end{vmatrix}, \quad \delta_3 = -\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_4 & y_4 \end{vmatrix}, \quad \delta_4 = \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix} \tag{4.10}$$

$$6V = \begin{vmatrix} 1 & x_i & y_i & z_i \\ 1 & x_j & y_j & z_j \\ 1 & x_k & y_k & z_k \\ 1 & x_l & y_l & z_l \end{vmatrix} \tag{4.11}$$

Because of the differentials in strain calculation, α is not used in the following stages. If we expand Equation 4.6, we obtain

$$u^e(x, y, z) = \frac{1}{6V^e} \begin{bmatrix} \alpha_1 + \beta_1 x + \gamma_1 y + \delta_1 z \\ \alpha_2 + \beta_2 x + \gamma_2 y + \delta_2 z \\ \alpha_3 + \beta_3 x + \gamma_3 y + \delta_3 z \\ \alpha_4 + \beta_4 x + \gamma_4 y + \delta_4 z \end{bmatrix} \begin{bmatrix} u(x,y,z)_1 & u(x,y,z)_2 & u(x,y,z)_3 & u(x,y,z)_4 \end{bmatrix} \tag{4.12}$$

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

$$\begin{aligned}
\psi_1 &= \tfrac{1}{6V}\left(\alpha_1 + \beta_1 x + \gamma_1 y + \delta_1 z\right) u(x,y,z)_1 \\
\psi_2 &= \tfrac{1}{6V}\left(\alpha_2 + \beta_2 x + \gamma_2 y + \delta_2 z\right) u(x,y,z)_2 \\
\psi_3 &= \tfrac{1}{6V}\left(\alpha_3 + \beta_3 x + \gamma_3 y + \delta_3 z\right) u(x,y,z)_3 \\
\psi_4 &= \tfrac{1}{6V}\left(\alpha_4 + \beta_4 x + \gamma_4 y + \delta_4 z\right) u(x,y,z)_4
\end{aligned} \tag{4.13}$$

The next step is to find the infinitesimal strains that are used to calculate the global stiffness matrix. Figure 4.2 shows that the element edge that lies on x-axis AB becomes $A'B'$. The engineering normal strain is calculated as the change in the length of the line [?]:

Figure 4.2: 2D element before and after deformation [?].

$$\varepsilon_x = \frac{A'B' - AB}{AB} \tag{4.14}$$

and

$$AB = dx. \tag{4.15}$$

The elemental edge dx that is initially parallel to the x-axis is deformed as $(A'B')^2$ (cf. Equation 4.16). The final length can be evaluated by

$$\begin{aligned}
(A'B')^2 &= \left(dx + \tfrac{\partial u_x}{\partial x} dx\right)^2 + \left(\tfrac{\partial u_y}{\partial x} dx\right)^2, \\
(A'B')^2 &= dx\left[1 + 2\left(\tfrac{\partial u_x}{\partial x}\right) + \left(\tfrac{\partial u_x}{\partial x}\right)^2 + \left(\tfrac{\partial u_y}{\partial x}\right)^2\right].
\end{aligned} \tag{4.16}$$

By neglecting the higher order terms in Equation 4.16, the 2D strains are defined by

$$\begin{aligned}
\varepsilon_{xx} &= \tfrac{\partial u_x}{\partial x} \\
\varepsilon_{yy} &= \tfrac{\partial u_y}{\partial y} \\
\gamma_{xy} &= \tfrac{1}{2}\left(\tfrac{\partial u_x}{\partial y} + \tfrac{\partial u_y}{\partial x}\right)
\end{aligned} \tag{4.17}$$

After finding the 2D strains, it is straightforward to expand it to the 3D case by

$$\begin{aligned}
\varepsilon_{zz} &= \tfrac{\partial u_z}{\partial z} \\
\gamma_{zx} &= \tfrac{1}{2}\left(\tfrac{\partial u_z}{\partial x} + \tfrac{\partial u_x}{\partial z}\right) \\
\gamma_{yz} &= \tfrac{1}{2}\left(\tfrac{\partial u_y}{\partial z} + \tfrac{\partial u_z}{\partial y}\right)
\end{aligned} \tag{4.18}$$

Infinitesimal strains with 3D element are given by

$$\{\varepsilon\} = \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ 2\gamma_{xy} \\ 2\gamma_{zx} \\ 2\gamma_{yz} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u_x}{\partial x} \\ \frac{\partial u_y}{\partial y} \\ \frac{\partial u_z}{\partial z} \\ \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\ \frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z} \\ \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \end{Bmatrix}. \tag{4.19}$$

After finding strains, these equations are combined with shape functions to find matrix $[B]$:

$$\{\varepsilon\} = [B]\{d\}. \tag{4.20}$$

Using Equations 4.13 for displacements, we can evaluate the partial derivatives of the shape functions as follows:

$$\begin{aligned}
\frac{\partial u_x}{\partial x} &= \frac{\partial}{\partial x}(\psi_1+\psi_2+\psi_3+\psi_4) = \frac{1}{6V}(\beta_1 u_1+\beta_2 u_2+\beta_3 u_3+\beta_4 u_4)\\
\frac{\partial u_y}{\partial y} &= \frac{1}{6V}(\gamma_1 v_1+\gamma_2 v_2+\gamma_3 v_3+\gamma_4 v_4)\\
\frac{\partial u_z}{\partial z} &= \frac{1}{6V}(\delta_1 w_1+\delta_2 w_2+\delta_3 w_3+\delta_4 w_4)\\
\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x} &= \frac{1}{6V}(\gamma_1 u_1+\gamma_2 u_2+\gamma_3 u_3+\gamma_4 u_4+\beta_1 v_1+\beta_2 v_2+\beta_3 v_3+\beta_4 v_4)\\
\frac{\partial u_z}{\partial x}+\frac{\partial u_x}{\partial z} &= \frac{1}{6V}(\beta_1 w_1+\beta_2 w_2+\beta_3 w_3+\beta_4 w_4+\delta_1 u_1+\delta_2 u_2+\delta_3 u_3+\delta_4 u_4)\\
\frac{\partial u_y}{\partial z}+\frac{\partial u_z}{\partial y} &= \frac{1}{6V}(\delta_1 v_1+\delta_2 v_2+\delta_3 v_3+\delta_4 v_4+\gamma_1 w_1+\gamma_2 w_2+\gamma_3 w_3+\gamma_4 w_4)
\end{aligned} \tag{4.21}$$

Using Equations 4.21 for the 1^{st} node, we obtain the submatrix $[B_1]$ of $[B]$ as

$$[B_1] = \begin{bmatrix} \frac{\partial u_x}{\partial x} & 0 & 0 \\ 0 & \frac{\partial u_y}{\partial y} & 0 \\ 0 & 0 & \frac{\partial u_z}{\partial z} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial x} & 0 \\ \frac{\partial u_z}{\partial x} & 0 & \frac{\partial u_x}{\partial z} \\ 0 & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial y} \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \end{Bmatrix}. \tag{4.22}$$

Using Equations 4.21 and 4.22, $\{\varepsilon\}$ can be written as

$$\{\varepsilon\} = [B]\{d\} = \begin{bmatrix} \beta_1 & 0 & 0 & \beta_2 & 0 & 0 & \beta_3 & 0 & 0 & \beta_4 & 0 & 0 \\ 0 & \gamma_1 & 0 & 0 & \gamma_2 & 0 & 0 & \gamma_3 & 0 & 0 & \gamma_4 & 0 \\ 0 & 0 & \delta_1 & 0 & 0 & \delta_2 & 0 & 0 & \delta_3 & 0 & 0 & \delta_4 \\ \gamma_1 & \beta_1 & 0 & \gamma_2 & \beta_2 & 0 & \gamma_3 & \beta_3 & 0 & \gamma_4 & \beta_4 & 0 \\ \delta_1 & 0 & \beta_1 & \delta_2 & 0 & \beta_2 & \delta_3 & 0 & \beta_3 & \delta_4 & 0 & \beta_4 \\ 0 & \delta_1 & \gamma_1 & 0 & \delta_2 & \gamma_2 & 0 & \delta_3 & \gamma_3 & 0 & \delta_5 & \gamma_5 \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \\ u_2 \\ v_2 \\ w_2 \\ u_3 \\ v_3 \\ w_3 \\ u_4 \\ v_4 \\ w_4 \end{Bmatrix}. \tag{4.23}$$

In Equation **??**, the engineering stress vector $\{\sigma\}$ is related to the strain vector $\{\varepsilon\}$ by

$$\begin{aligned} \{\sigma\} &= [E]\{\varepsilon\}, \\ \{\sigma\} &= [E][B]\{d\}, \end{aligned} \tag{4.24}$$

where $[E]$ is the material property matrix (constitutive matrix) defined by

$$[E] = \frac{\epsilon}{(1+\nu)(1-2\nu)} \begin{bmatrix} (1-\nu) & \nu & \nu & 0 & 0 & 0 \\ \nu & (1-\nu) & \nu & 0 & 0 & 0 \\ \nu & \nu & (1-\nu) & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(1-2\nu)}{2} \end{bmatrix}, \tag{4.25}$$

where ϵ is the Young's modulus and ν is the Poisson's ratio. Young's modulus describes the elastic properties of a solid undergoing tension or compression. Poisson's ratio is the ratio of transverse strain (perpendicular to the applied load), to the longitudinal strain (in the direction of the applied load) [**?**]. From the conservation of the potential energy, substituting Equations 4.20 and 4.24 into Equation **??**, we obtain the element stiffness matrix

$$[k] = \int\int\int \{d\}^T[B]^T[E][B]\{d\}dx\ dy\ dz. \tag{4.26}$$

As seen from Equations 4.22 and 4.25, the matrices $[B]$ and $[E]$ are constant for a tetrahedral element, so that Equation 4.26 is rewritten as

$$[k] = \{d\}^T[B]^T[E][B]\{d\}V. \tag{4.27}$$

With the introduction of the nodal forces per element,

$$\left\{ f \right\} = \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{1z} \\ \vdots \\ f_{4x} \\ f_{4y} \\ f_{4z} \end{Bmatrix} \{d\}^T. \tag{4.28}$$

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

$$K^e\{d\}^e = \{f\}^e. \tag{4.29}$$

By substituting $\{d\}$ with u, we obtain [?]:

$$K^e u^e = f^e. \tag{4.30}$$

4.1.3 Assembly of Elemental Stiffness Matrices

□

Figure 4.3: A tetrahedral mesh with two elements

We apply assembly process using the element's nodal information (which nodes belong to which elements). The size of elemental stiffness matrix is 12×12 (the tetrahedron has four nodes and it is three-dimensional). The size of global stiffness matrix is $3N \times 3N$, where N is the total number of nodes of the whole system. In order to complete the assembly process, all elemental stiffness matrices must be copied to the correct index of the global stiffness matrix. The assembly operation is described by using two elements (Figure 4.3) as an example. The stiffness matrices of the first and second elements are given as

$$K_1 = \begin{bmatrix} k^1_{1,1} & k^1_{1,2} & k^1_{1,3} & \dots & k^1_{1,12} \\ k^1_{2,1} & k^1_{2,2} & k^1_{2,3} & \dots & k^1_{2,12} \\ k^1_{3,1} & k^1_{3,2} & k^1_{3,3} & \dots & k^1_{3,12} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k^1_{12,1} & k^1_{12,2} & k^1_{12,3} & \dots & k^1_{12,12} \end{bmatrix} \text{ and } K_2 = \begin{bmatrix} k^2_{1,1} & k^2_{1,2} & k^2_{1,3} & \dots & k^2_{1,12} \\ k^2_{2,1} & k^2_{2,2} & k^2_{2,3} & \dots & k^2_{2,12} \\ k^2_{3,1} & k^2_{3,2} & k^2_{3,3} & \dots & k^2_{3,12} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k^2_{12,1} & k^2_{12,2} & k^2_{12,3} & \dots & k^2_{12,12} \end{bmatrix}. \tag{4.31}$$

It can be seen from Figure 4.3 that Nodes 2, 3 and 4 belong to both Element 1 and Element 2. When we assemble the elements, these shared values in the global stiffness matrix (5 nodes, 15×15 matrix) come from both Element 1 and Element 2. The elements are assembled using Algorithm 1, which constructs the global stiffness matrix K (see Equation 4.32).

Algorithm 1 Assembly of the Elements

---

for i = 1 to N **do**
- f_{BI} = ((1^{st} node of i^{th} element - 1) $\times$ 3) + 1
- s_{BI} = ((2^{nd} node of i^{th} element - 1) $\times$ 3) + 1
- t_{BI} = ((3^{rd} node of i^{th} element - 1) $\times$ 3) + 1
- r_{BI} = ((4^{th} node of i^{th} element - 1) $\times$ 3) + 1
- $K[f_{BI}{:}f_{BI}{+}2,\ f_{BI}{:}\ f_{BI}{+}2]$ += K_i[1:3,1:3]
- $K[f_{BI}{:}f_{BI}{+}2,\ s_{BI}{:}\ s_{BI}{+}2]$ += K_i[1:3,4:6]
- $K[f_{BI}{:}f_{BI}{+}2,\ t_{BI}{:}\ t_{BI}{+}2]$ += K_i[1:3,7:9]
- $K[f_{BI}{:}f_{BI}{+}2,\ r_{BI}{:}\ r_{BI}{+}2]$ = K_i[1:3,10:12]
- $K[s_{BI}{:}s_{BI}{+}2,\ f_{BI}{:}\ f_{BI}{+}2]$ += K_i[4:6,1:3]
- $K[s_{BI}{:}s_{BI}{+}2,\ s_{BI}{:}\ s_{BI}{+}2]$ += K_i[4:6,4:6]
- $K[s_{BI}{:}s_{BI}{+}2,\ t_{BI}{:}\ t_{BI}{+}2]$ += K_i[4:6,7:9]
- $K[s_{BI}{:}s_{BI}{+}2,\ r_{BI}{:}\ r_{BI}{+}2]$ += K_i[4:6,10:12]
- $K[t_{BI}{:}t_{BI}{+}2,\ f_{BI}{:}\ f_{BI}{+}2]$ += K_i[7:9,1:3]
- $K[t_{BI}{:}t_{BI}{+}2,\ s_{BI}{:}\ s_{BI}{+}2]$ += K_i[7:9,4:6]
- $K[t_{BI}{:}t_{BI}{+}2,\ t_{BI}{:}\ t_{BI}{+}2]$ += K_i[7:9,7:9]
- $K[t_{BI}{:}t_{BI}{+}2,\ r_{BI}{:}\ r_{BI}{+}2]$ += K_i[7:9,10:12]
- $K[r_{BI}{:}r_{BI}{+}2,\ r_{BI}{:}\ r_{BI}{+}2]$ += K_i[10:12,1:3]
- $K[r_{BI}{:}r_{BI}{+}2,\ s_{BI}{:}\ s_{BI}{+}2]$ += K_i[10:12,4:6]
- $K[r_{BI}{:}r_{BI}{+}2,\ t_{BI}{:}\ t_{BI}{+}2]$ += K_i[10:12,7:9]
- $K[r_{BI}{:}r_{BI}{+}2,\ r_{BI}{:}\ r_{BI}{+}2]$ += K_i[10:12,10:12]

end for

$$K = \left[\begin{array}{ccccccccccccccc}
k^1_{1,1} & k^1_{1,2} & k^1_{1,3} & k^1_{1,4} & k^1_{1,5} & k^1_{1,6} & k^1_{1,7} & k^1_{1,8} & k^1_{1,9} & k^1_{1,10} & k^1_{1,11} & k^1_{1,12} & 0 & 0 & 0 \\
k^1_{2,1} & k^1_{2,2} & k^1_{2,3} & k^1_{2,4} & k^1_{2,5} & k^1_{2,6} & k^1_{2,7} & k^1_{2,8} & k^1_{2,9} & k^1_{2,10} & k^1_{2,11} & k^1_{2,12} & 0 & 0 & 0 \\
k^1_{3,1} & k^1_{3,2} & k^1_{3,3} & k^1_{3,4} & k^1_{3,5} & k^1_{3,6} & k^1_{3,7} & k^1_{3,8} & k^1_{3,9} & k^1_{3,10} & k^1_{3,11} & k^1_{3,12} & 0 & 0 & 0 \\
k^1_{4,1} & k^1_{4,2} & k^1_{4,3} & k^1_{4,4}+k^2_{1,1} & k^1_{4,5}+k^2_{1,2} & k^1_{4,6}+k^2_{1,3} & k^1_{4,7}+k^2_{1,4} & k^1_{4,8}+k^2_{1,5} & k^1_{4,9}+k^2_{1,6} & k^1_{4,10}+k^2_{1,7} & k^1_{4,11}+k^2_{1,8} & k^1_{4,12}+k^2_{1,9} & k^2_{1,10} & k^2_{1,11} & k^2_{1,12} \\
k^1_{5,1} & k^1_{5,2} & k^1_{5,3} & k^1_{5,4}+k^2_{2,1} & k^1_{5,5}+k^2_{2,2} & k^1_{5,6}+k^2_{2,3} & k^1_{5,7}+k^2_{2,4} & k^1_{5,8}+k^2_{2,5} & k^1_{5,9}+k^2_{2,6} & k^1_{5,10}+k^2_{2,7} & k^1_{5,11}+k^2_{2,8} & k^1_{5,12}+k^2_{2,9} & k^2_{2,10} & k^2_{2,11} & k^2_{2,12} \\
k^1_{6,1} & k^1_{6,2} & k^1_{6,3} & k^1_{6,4}+k^2_{3,1} & k^1_{6,5}+k^2_{3,2} & k^1_{6,6}+k^2_{3,3} & k^1_{6,7}+k^2_{3,4} & k^1_{6,8}+k^2_{3,5} & k^1_{6,9}+k^2_{3,6} & k^1_{6,10}+k^2_{3,7} & k^1_{6,11}+k^2_{3,8} & k^1_{6,12}+k^2_{3,9} & k^2_{3,10} & k^2_{3,11} & k^2_{3,12} \\
k^1_{7,1} & k^1_{7,2} & k^1_{7,3} & k^1_{7,4}+k^2_{4,1} & k^1_{7,5}+k^2_{4,2} & k^1_{7,6}+k^2_{4,3} & k^1_{7,7}+k^2_{4,4} & k^1_{7,8}+k^2_{4,5} & k^1_{7,9}+k^2_{4,6} & k^1_{7,10}+k^2_{4,7} & k^1_{7,11}+k^2_{4,8} & k^1_{7,12}+k^2_{4,9} & k^2_{4,10} & k^2_{4,11} & k^2_{4,12} \\
k^1_{8,1} & k^1_{8,2} & k^1_{8,3} & k^1_{8,4}+k^2_{5,1} & k^1_{8,5}+k^2_{5,2} & k^1_{8,6}+k^2_{5,3} & k^1_{8,7}+k^2_{5,4} & k^1_{8,8}+k^2_{5,5} & k^1_{8,9}+k^2_{5,6} & k^1_{8,10}+k^2_{5,7} & k^1_{8,11}+k^2_{5,8} & k^1_{8,12}+k^2_{5,9} & k^2_{5,10} & k^2_{5,11} & k^2_{5,12} \\
k^1_{9,1} & k^1_{9,2} & k^1_{9,3} & k^1_{9,4}+k^2_{6,1} & k^1_{9,5}+k^2_{6,2} & k^1_{9,6}+k^2_{6,3} & k^1_{9,7}+k^2_{6,4} & k^1_{9,8}+k^2_{6,5} & k^1_{9,9}+k^2_{6,6} & k^1_{9,10}+k^2_{6,7} & k^1_{9,11}+k^2_{6,8} & k^1_{9,12}+k^2_{6,9} & k^2_{6,10} & k^2_{6,11} & k^2_{6,12} \\
k^1_{10,1} & k^1_{10,2} & k^1_{10,3} & k^1_{10,4}+k^2_{7,1} & k^1_{10,5}+k^2_{7,2} & k^1_{10,6}+k^2_{7,3} & k^1_{10,7}+k^2_{7,4} & k^1_{10,8}+k^2_{7,5} & k^1_{10,9}+k^2_{7,6} & k^1_{10,10}+k^2_{7,7} & k^1_{10,11}+k^2_{7,8} & k^1_{10,12}+k^2_{7,9} & k^2_{7,10} & k^2_{7,11} & k^2_{7,12} \\
k^1_{11,1} & k^1_{11,2} & k^1_{11,3} & k^1_{11,4}+k^2_{8,1} & k^1_{11,5}+k^2_{8,2} & k^1_{11,6}+k^2_{8,3} & k^1_{11,7}+k^2_{8,4} & k^1_{11,8}+k^2_{8,5} & k^1_{11,9}+k^2_{8,6} & k^1_{11,10}+k^2_{8,7} & k^1_{11,11}+k^2_{8,8} & k^1_{11,12}+k^2_{8,9} & k^2_{8,10} & k^2_{8,11} & k^2_{8,12} \\
k^1_{12,1} & k^1_{12,2} & k^1_{12,3} & k^1_{12,4}+k^2_{9,1} & k^1_{12,5}+k^2_{9,2} & k^1_{12,6}+k^2_{9,3} & k^1_{12,7}+k^2_{9,4} & k^1_{12,8}+k^2_{9,5} & k^1_{12,9}+k^2_{9,6} & k^1_{12,10}+k^2_{9,7} & k^1_{12,11}+k^2_{9,8} & k^1_{12,12}+k^2_{9,9} & k^2_{9,10} & k^2_{9,11} & k^2_{9,12} \\
0 & 0 & 0 & k^2_{10,1} & k^2_{10,2} & k^2_{10,3} & k^2_{10,4} & k^2_{10,5} & k^2_{10,6} & k^2_{10,7} & k^2_{10,8} & k^2_{10,9} & k^2_{10,10} & k^2_{10,11} & k^2_{10,12} \\
0 & 0 & 0 & k^2_{11,1} & k^2_{11,2} & k^2_{11,3} & k^2_{11,4} & k^2_{11,5} & k^2_{11,6} & k^2_{11,7} & k^2_{11,8} & k^2_{11,9} & k^2_{11,10} & k^2_{11,11} & k^2_{11,12} \\
0 & 0 & 0 & k^2_{12,1} & k^2_{12,2} & k^2_{12,3} & k^2_{12,4} & k^2_{12,5} & k^2_{12,6} & k^2_{12,7} & k^2_{12,8} & k^2_{12,9} & k^2_{12,10} & k^2_{12,11} & k^2_{12,12}
\end{array}\right]. \tag{4.32}$$

4.1.4 Applying Boundary Conditions

After assembling the elemental stiffness matrices and nodal force vectors, boundary conditions are applied by assigning 1*s* and 0*s* to the corresponding rows and columns according to constrained nodes by Algorithm 2.

Algorithm 2 Boundary Value Assignment

---

for i = 1 to BC (BC is the number of constrained nodes) **do**
    BI is ((i^{th} constrained node - 1) × 3) + 1
    K[BI: BI + 2, 1: dimension] = 0
    K[1: dimension, BI: BI + 2] = 0
    K[BI, BI] = 1
    K[BI+1, BI+1] = 1
    K[BI+2, BI+2] = 1
    F[BI: BI + 2] = 0
end for

The system in Figure 4.3 is constrained from nodes 1, 2, and 3. Algorithm 2 is used to obtain the global stiffness matrix K:

$$K = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^1_{10,10}+k^2_{7,7}k^1_{10,10}+k^2_{7,7} & k^1_{10,11}+k^2_{7,8} & k^1_{10,12}+k^2_{7,9} & k^2_{7,10} & k^2_{7,11} & k^2_{7,12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^1_{11,10}+k^2_{8,7} & k^1_{11,11}+k^2_{8,8} & k^1_{11,12}+k^2_{8,9} & k^2_{8,10} & k^2_{8,11} & k^2_{8,12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^1_{12,10}+k^2_{9,7} & k^1_{12,11}+k^2_{9,8} & k^1_{12,12}+k^2_{9,9} & k^2_{9,10} & k^2_{9,11} & k^2_{9,12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^2_{10,7} & k^2_{10,8} & k^2_{10,9} & k^2_{10,10} & k^2_{10,11} & k^2_{10,12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^2_{11,7} & k^2_{11,8} & k^2_{11,9} & k^2_{11,10} & k^2_{11,11} & k^2_{11,12} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & k^2_{12,7} & k^2_{12,8} & k^2_{12,9} & k^2_{12,10} & k^2_{12,11} & k^2_{12,12}
\end{bmatrix}. \tag{4.33}$$

4.1.5 Solution of the Linear System

After applying boundary conditions to the elemental stiffness matrices and nodal force vectors, the whole system is one large linear system:

$$Ku = f. \tag{4.34}$$

In the last step, solving the system gives unknown nodal displacements

$$u = K^{-1} f. \tag{4.35}$$

Chapter 5

Non-Linear Finite Element Method

This chapter explains in detail the stages of the nonlinear FEM and a verification procedure for measuring the correctness of the proposed nonlinear FEM.

5.1 Non-Linear FEM using Tetrahedral Elements with Green-Lagrange Strains

Proposed system uses non-linear FEM due to accuracy reasons. In this chapter, algorithm of non-linear FEM solution, the development of Green-Lagrange strains (large deformation strains) η that leads to non-linear FEM, stiffness matrix K and the solution of the system with Newton-Raphson method will be explained. Our proposed non-linear FEM solution algorithm for 3D tetrahedral element consist of 4 main parts:

1. Tetrahedralization;

2. Construction of nonlinear elemental stiffness matrices;

3. Construction of nonlinear element residuals;
4. Solution of the non-linear system with Newton-Raphson method that gives unknown nodal displacements.

The proposed nonlinear FEM uses linear FEM's style in the sense that it does not require the explicit use of weight functions, differential equations and integrals. Moreover, our approach extends the linear FEM to the nonlinear FEM by extending the linear strains to the Green-Lagrange strains.

5.1.1 Tetrahedralization

We have the information of nodal positions and which nodes belong to which element from the tetrahedralization process like we did for linear FEM solution. We use nodal positions to construct elemental stiffness matrices, and element's nodal information to assemble the element's Jacobian matrices to form global Jacobian matrices and element's residuals to form global residual vectors for every step of Newton-Raphson process.

5.1.2 Construction of Nonlinear Elemental Stiffness Matrices

The displacements are represented with linear shape functions, as in linear FEM. The calculation of the parameters α, β, γ, δ, V (the volume of tetrahedron), and the shape functions are the same as it is done in linear FEM.

Nonlinear FEM differs from linear FEM because of the nonlinearity that arises from the higher order term neglected in Equation 4.16. The strain vector that is used in linear FEM relies on the assumption that the displacements at x, y and z axes are very small. The initial and final positions of a given particle are practically the same; thus, the higher terms are neglected [**?**]. When the displacements are large, however, this is no longer the case and one must distinguish between the initial and final coordinates of particles, so the higher order terms are added into the strain equations.

$$(A'B')^2 = \left(dx + \frac{\partial u_x}{\partial x}dx\right)^2 + \left(\frac{\partial u_y}{\partial x}dx\right)^2. \tag{5.1}$$

By adding the higher order terms, 3D strains are defined as [**?**]:

$$\begin{aligned}
\eta_{xx} &= \tfrac{\partial u_x}{\partial x} + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial x}\tfrac{\partial u_x}{\partial x}\right) + \left(\tfrac{\partial u_y}{\partial x}\tfrac{\partial u_y}{\partial x}\right) + \left(\tfrac{\partial u_z}{\partial x}\tfrac{\partial u_z}{\partial x}\right)\right] \\
\eta_{yy} &= \tfrac{\partial u_y}{\partial y} + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial y}\tfrac{\partial u_x}{\partial y}\right) + \left(\tfrac{\partial u_y}{\partial y}\tfrac{\partial u_y}{\partial y}\right) + \left(\tfrac{\partial u_z}{\partial y}\tfrac{\partial u_z}{\partial y}\right)\right] \\
\eta_{zz} &= \tfrac{\partial u_z}{\partial z} + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial z}\tfrac{\partial u_x}{\partial z}\right) + \left(\tfrac{\partial u_y}{\partial z}\tfrac{\partial u_y}{\partial z}\right) + \left(\tfrac{\partial u_z}{\partial z}\tfrac{\partial u_z}{\partial z}\right)\right] \\
\gamma_{xy} &= \tfrac{1}{2}\left(\tfrac{\partial u_x}{\partial y} + \tfrac{\partial u_y}{\partial x}\right) + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial x}\tfrac{\partial u_x}{\partial y}\right) + \left(\tfrac{\partial u_y}{\partial x}\tfrac{\partial u_y}{\partial y}\right) + \left(\tfrac{\partial u_z}{\partial x}\tfrac{\partial u_z}{\partial y}\right)\right] \\
\gamma_{zx} &= \tfrac{1}{2}\left(\tfrac{\partial u_z}{\partial x} + \tfrac{\partial u_x}{\partial z}\right) + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial z}\tfrac{\partial u_x}{\partial x}\right) + \left(\tfrac{\partial u_y}{\partial z}\tfrac{\partial u_y}{\partial x}\right) + \left(\tfrac{\partial u_z}{\partial z}\tfrac{\partial u_z}{\partial x}\right)\right] \\
\gamma_{yz} &= \tfrac{1}{2}\left(\tfrac{\partial u_y}{\partial z} + \tfrac{\partial u_z}{\partial y}\right) + \tfrac{1}{2}\left[\left(\tfrac{\partial u_x}{\partial y}\tfrac{\partial u_x}{\partial z}\right) + \left(\tfrac{\partial u_y}{\partial y}\tfrac{\partial u_y}{\partial z}\right) + \left(\tfrac{\partial u_z}{\partial y}\tfrac{\partial u_z}{\partial z}\right)\right]
\end{aligned} \tag{5.2}$$

that leads to

$$\{\eta\} = \begin{Bmatrix} \eta_{xx} \\ \eta_{yy} \\ \eta_{zz} \\ 2\left(\gamma_{xy} + \gamma_{yx}\right) \\ 2\left(\gamma_{xz} + \gamma_{zx}\right) \\ 2\left(\gamma_{yz} + \gamma_{zy}\right) \end{Bmatrix} = \begin{Bmatrix} \eta_{xx} \\ \eta_{yy} \\ \eta_{zz} \\ 2\eta_{xy} \\ 2\eta_{zx} \\ 2\eta_{yz} \end{Bmatrix}. \tag{5.3}$$

Green-Lagrange strain tensor is represented in matrix notation as

$$\{\eta\} = [B_L]\{d\} + \frac{1}{2}\{d\}^T[B_{NL}]\{d\}, \tag{5.4}$$

where $\{d\}$ is the nodal displacements, $[B_L]$ is the linear and $[B_{NL}]$ is the nonlinear part of the $[B_0]$ matrix [?]. For a specific element, $[B_L]$ and $[B_{NL}]$ are constant, as the $[B]$ matrix in linear FEM. With the modification of $\{d\}$ by introducing secant and tangent relations [?], Equation 5.4 becomes

$$\begin{aligned} \{\eta\} &= \left([B_L] + \tfrac{1}{2}\{d^T\}[B_{NL}]\right)\{d\} = [B_0]\{d\}, \\ \{\bar{\eta}\} &= \left([B_L] + \{d^T\}[B_{NL}]\right)\{d\} = [\bar{B}_0]\{d\}. \end{aligned} \tag{5.5}$$

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

$$\{\bar{\eta}\} = [\bar{B}_0]\{d\}. \tag{5.6}$$

The most frequently used terms for the calculation of the nonlinear strains are $\frac{\partial u_x}{\partial x}$, $\frac{\partial u_x}{\partial y}$, $\frac{\partial u_x}{\partial z}$, $\frac{\partial u_y}{\partial x}$, $\frac{\partial u_y}{\partial y}$, $\frac{\partial u_y}{\partial x}$, $\frac{\partial u_z}{\partial x}$, $\frac{\partial u_z}{\partial y}$ and $\frac{\partial u_z}{\partial z}$. They are represented by

$$\begin{aligned} u_{xx} &= (\beta_1 u_1 + \beta_2 u_2 + \beta_3 u_3 + \beta_4 u_4) \\ u_{yx} &= (\beta_1 v_1 + \beta_2 v_2 + \beta_3 v_3 + \beta_4 v_4) \\ u_{zx} &= (\beta_1 w_1 + \beta_2 w_2 + \beta_3 w_3 + \beta_4 w_4) \\ u_{xy} &= (\gamma_1 u_1 + \gamma_2 u_2 + \gamma_3 u_3 + \gamma_4 u_4) \\ u_{yy} &= (\gamma_1 v_1 + \gamma_2 v_2 + \gamma_3 v_3 + \gamma_4 v_4) \\ u_{zy} &= (\gamma_1 w_1 + \gamma_2 w_2 + \gamma_3 w_3 + \gamma_4 w_4) \\ u_{xz} &= (\delta_1 u_1 + \delta_2 u_2 + \delta_3 u_3 + \delta_4 u_4) \\ u_{yz} &= (\delta_1 v_1 + \delta_2 v_2 + \delta_3 v_3 + \delta_4 v_4) \\ u_{zz} &= (\delta_1 w_1 + \delta_2 w_2 + \delta_3 w_3 + \delta_4 w_4) \end{aligned} \tag{5.7}$$

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

Using the linear parts of Equation 4.13 for displacements, we can evaluate the partial derivatives of the shape functions as follows (for the 1^{st} node of $[B_{NL}]$):

$$
\begin{aligned}
\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial x}\right)+\left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial x}\right)+\left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial x}\right)\right] &= \tfrac{1}{6V}(\beta_1(u_{xx}+u_{yx}+u_{zx}))\\
\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial y}\right)+\left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial y}\right)+\left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial y}\right)\right] &= \tfrac{1}{6V}(\gamma_1(u_{xy}+u_{yy}+u_{zy}))\\
\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial z}\right)+\left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial z}\right)+\left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial z}\right)\right] &= \tfrac{1}{6V}(\delta_1(u_{xz}+u_{yz}+u_{zz}))\\
\left[\left(\frac{\partial u_x}{\partial x}\frac{\partial u_x}{\partial y}\right)+\left(\frac{\partial u_y}{\partial x}\frac{\partial u_y}{\partial y}\right)+\left(\frac{\partial u_z}{\partial x}\frac{\partial u_z}{\partial y}\right)\right] &= \tfrac{1}{6V}(\gamma_1(u_{xx}+u_{yx}+u_{zx}))+\\
&\quad \tfrac{1}{6V}(\beta_1(u_{xy}+u_{yy}+u_{zy}))\\
\left[\left(\frac{\partial u_x}{\partial z}\frac{\partial u_x}{\partial x}\right)+\left(\frac{\partial u_y}{\partial z}\frac{\partial u_y}{\partial x}\right)+\left(\frac{\partial u_z}{\partial z}\frac{\partial u_z}{\partial x}\right)\right] &= \tfrac{1}{6V}(\delta_1(u_{xx}+u_{yx}+u_{zx}))+\\
&\quad \tfrac{1}{6V}(\beta_1(u_{xz}+u_{yz}+u_{zz}))\\
\left[\left(\frac{\partial u_x}{\partial y}\frac{\partial u_x}{\partial z}\right)+\left(\frac{\partial u_y}{\partial y}\frac{\partial u_y}{\partial z}\right)+\left(\frac{\partial u_z}{\partial y}\frac{\partial u_z}{\partial z}\right)\right] &= \tfrac{1}{6V}(\gamma_1(u_{xz}+u_{yz}+u_{zz}))+\\
&\quad \tfrac{1}{6V}(\delta_1(u_{xy}+u_{yy}+u_{zy}))
\end{aligned}
\tag{5.8}
$$

Using Equations 5.5 and 5.8, we obtain $[\bar{B}_0]$ for the 1^{st} node

$$
[\bar{B}_{0_1}] = \begin{bmatrix}
\beta_1+\beta_1(u_{xx}) & \beta_1(u_{yx}) & \beta_1(u_{zx})\\
\gamma_1(u_{xy}) & \gamma_1+\gamma_1(u_{yy}) & \gamma_1(u_{zy})\\
\delta_1(u_{xz}) & \delta_1(u_{yz}) & \delta_1+\delta_1(u_{zz})\\
\gamma_1+\gamma_1(u_{xx})+\beta_1(u_{xy}) & \gamma_1(u_{yx})+\beta_1+\beta_1(u_{yy}) & \gamma_1(u_{zx})+\beta_1(u_{zy})\\
\delta_1+\delta_1(u_{xx})+\beta_1(u_{xz}) & \delta_1(u_{yx})+\beta_1(u_{yz}) & \delta_1(u_{zx})+\beta_1+\beta_1(u_{zz})\\
\gamma_1(u_{xz})+\delta_1(u_{xy}) & \gamma_1+\gamma_1(u_{yz})+\delta_1(u_{yy}) & \gamma_1(u_{zz})+\delta_1+\delta_1(u_{zy})
\end{bmatrix}
\begin{Bmatrix} u_1\\ v_1\\ w_1 \end{Bmatrix}.
\tag{5.9}
$$

Similarly, using Equations 5.5 and 5.8, we obtain $[B_0]$ for the 1^{st} node

$$[B_{0_1}] = \begin{bmatrix} \beta_1 + \frac{1}{2}\beta_1(u_{xx}) & \frac{1}{2}\beta_1(u_{yx}) & \frac{1}{2}\beta_1(u_{zx}) \\ \frac{1}{2}\gamma_1(u_{xy}) & \gamma_1 + \frac{1}{2}\gamma_1(u_{yy}) & \frac{1}{2}\gamma_1(u_{zy}) \\ \frac{1}{2}\delta_1(u_{xz}) & \frac{1}{2}\delta_1(u_{yz}) & \delta_1 + \frac{1}{2}\delta_1(u_{zz}) \\ \gamma_1 + \frac{1}{2}\gamma_1(u_{xx}) + \frac{1}{2}\beta_1(u_{xy}) & \frac{1}{2}\gamma_1(u_{yx}) + \beta_1 + \frac{1}{2}\beta_1(u_{yy}) & \frac{1}{2}\gamma_1(u_{zx}) + \frac{1}{2}\beta_1(u_{zy}) \\ \delta_1 + \frac{1}{2}\delta_1(u_{xx}) + \frac{1}{2}\beta_1(u_{xz}) & \frac{1}{2}\delta_1(u_{yx}) + \frac{1}{2}\beta_1(u_{yz}) & \frac{1}{2}\delta_1(u_{zx}) + \beta_1 + \frac{1}{2}\beta_1(u_{zz}) \\ \frac{1}{2}\gamma_1(u_{xz}) + \frac{1}{2}\delta_1(u_{xy}) & \gamma_1 + \frac{1}{2}\gamma_1(u_{yz}) + \frac{1}{2}\delta_1(u_{yy}) & \frac{1}{2}\gamma_1(u_{zz}) + \delta_1 + \frac{1}{2}\delta_1(u_{zy}) \end{bmatrix} \begin{Bmatrix} u_1 \\ v_1 \\ w_1 \end{Bmatrix}. \tag{5.10}$$

From Equation **??**, the engineering stress vector τ is related to the strain vector by

$$\tau = [E]\{\bar{\eta}\} = [E][\bar{B}_0]\{d\}. \tag{5.11}$$

From the conservation of the potential energy, substituting Equations 5.4 and 5.11 into Equation **??**, we obtain the element stiffness matrix

$$[k(u)] = \int\int\int \{d\}^T[B_0]^T[E][\bar{B}_0]\{d\}dx\ dy\ dz. \tag{5.12}$$

We can discard the integrals as we did for linear FEM. $[B_0]$, $[E]$ and $[\bar{B}_0]$ are constant for tetrahedral element, so that Equation 5.12 is rewritten by

$$[k(u)] = \{d\}^T[B_0]^T[E][\bar{B}_0]\{d\}V. \tag{5.13}$$

Introducing nodal forces, we obtain

$$\left\{ f \right\} = \begin{Bmatrix} f_{1x} \\ f_{1y} \\ f_{1z} \\ \vdots \\ f_{4x} \\ f_{4y} \\ f_{4z} \end{Bmatrix} \{d\}^T. \tag{5.14}$$

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

$$k(d)^e \{d\}^e = f^e. \tag{5.15}$$

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

$$k(u)^e u^e = f^e. \tag{5.16}$$

Finally, there are only nonlinear displacement functions left, which are solved with the Newton-Raphson method to find the unknown displacements u.

5.1.3 Construction of Nonlinear Element Residuals

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

$$r^e = k(u)^e - f^e. \tag{5.17}$$

Having determined r^e, we can now express Equation 5.17 in expanded vector form as

$$\begin{Bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{12} \end{Bmatrix} = \begin{bmatrix} k(u)_{(1,1)} + k(u)_{(1,2)} + k(u)_{(1,3)} + \ldots + k(u)_{(1,12)} \\ k(u)_{(2,1)} + k(u)_{(2,2)} + k(u)_{(2,3)} + \ldots + k(u)_{(2,12)} \\ k(u)_{(3,1)} + k(u)_{(3,2)} + k(u)_{(3,3)} + \ldots + k(u)_{(3,12)} \\ \vdots \\ k(u)_{(12,1)} + k(u)_{(12,2)} + k(u)_{(12,3)} + \ldots + k(u)_{(12,12)} \end{bmatrix} - \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{12} \end{Bmatrix}. \tag{5.18}$$

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

$$r'^e = [K]_T^e = \begin{bmatrix} \frac{\partial}{\partial u_1} r_1 & \frac{\partial}{\partial v_1} r_1 & \frac{\partial}{\partial w_1} r_1 & \cdots & \frac{\partial}{\partial u_4} r_1 & \frac{\partial}{\partial v_4} r_1 & \frac{\partial}{\partial w_4} r_1 \\ \frac{\partial}{\partial u_1} r_2 & \frac{\partial}{\partial v_1} r_2 & \frac{\partial}{\partial w_1} r_2 & \cdots & \frac{\partial}{\partial u_4} r_2 & \frac{\partial}{\partial v_4} r_2 & \frac{\partial}{\partial w_4} r_2 \\ \frac{\partial}{\partial u_1} r_3 & \frac{\partial}{\partial v_1} r_3 & \frac{\partial}{\partial w_1} r_3 & \cdots & \frac{\partial}{\partial u_4} r_3 & \frac{\partial}{\partial v_4} r_3 & \frac{\partial}{\partial w_4} r_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial}{\partial u_1} r_{12} & \frac{\partial}{\partial v_1} r_{12} & \frac{\partial}{\partial w_1} r_{12} & \cdots & \frac{\partial}{\partial u_4} r_{12} & \frac{\partial}{\partial v_4} r_{12} & \frac{\partial}{\partial w_4} r_{12} \end{bmatrix}. \tag{5.19}$$

5.1.4 Solution of the Non-linear System with Newton-Raphson Method

Newton-Raphson method is a fast and popular numerical method for solving non-linear equations [**?**], as compared to the other methods, such as direct iteration. In principle, the method works by applying two steps (cf. Algorithm 3): (i) check if the equilibrium is reached within the desired accuracy; (ii) if not, make a suitable adjustment to the state of the deformation [**?**]. An initial guess for displacements are needed to start the iterations. The displacements are updated according to

$$x_{k+1} = x_k - \frac{f_{x_k}}{f'_{x_k}}. \tag{5.20}$$

Algorithm 3 Newton-Raphson method

Make initial guess $f(x)$
while $|f(x)| \leq \delta$ **do**
  Compute $p = -\frac{f(x)}{f'(x)}$
  Update $x = x + p$
  Calculate $f(x)$
end while

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

$$u_1 = u_0 - \frac{r_{u_0}}{r'_{u_0}}, \tag{5.21}$$

where r is residual of the global stiffness matrix $[K]$ calculated in Equation 5.18 and r' is the tangent stiffness matrix calculated in Equation 5.19.

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

$$\begin{aligned} &r'(u_i)p = -r(u_i), \quad \text{and} \\ &p = -(r'(u_i))^{-1}r(u_i). \end{aligned} \tag{5.22}$$

u_i is updated with the solution of Equation 5.22.

$$u_{i+1} = u_i + p. \tag{5.23}$$

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

$$|r(u_i)| \leq \delta. \tag{5.24}$$

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

5.2 Verification of the Proposed Approach

Verification is one of the important steps of the finite element analysis. We verified our approach with Pedersen's analytical stiffness matrices for tetrahedral elements solution [**?**]. In the experiments we obtained the same displacement amount with his method. In this section, Pedersen's method is explained in order to see the differences between our approach and his approach.

Both approaches give the same results since they use the same Green-Lagrange strains and tetrahedral elements. However, the computation times differ because of different methods to calculate the stiffness matrices. Pedersen divides the elemental stiffness matrices S into nine sub-matrices, $[S_{xx}]$, $[S_{xy}]$, $[S_{xz}]$, $[S_{yx}]$, $[S_{yy}]$, $[S_{yz}]$, $[S_{zx}]$, $[S_{zy}]$ and $[S_{zz}]$, which is represented as K (12×12 stiffness matrix) in our method.

$$
\begin{aligned}
S[1:4,1:4] &= [S_{xx}] \\
S[1:4,5:8] &= [S_{xy}] \\
S[1:4,9:12] &= [S_{xz}] \\
S[5:8,1:4] &= [S_{yx}] \\
S[5:8,5:8] &= [S_{yy}] \\
S[5:8,9:12] &= [S_{yz}] \\
S[9:12,1:4] &= [S_{zx}] \\
S[9:12,5:8] &= [S_{zy}] \\
S[9:12,9:12] &= [S_{zz}]
\end{aligned} \tag{5.25}
$$

These nine sub-matrices are calculated with 81 linear combination factors. Pedersen obtains $[S]_{xx}$ as

$$
\begin{aligned}
[S]_{xx} \quad = \quad & A_{xxxx}[T_{xx}] + A_{xxyy}[T_{yy}] + A_{xxzz}[T_{zz}] + A_{xxxy}[T_{xy}] + A_{xxyx}[T_{xy}^T] + \\
& A_{xxxz}[T_{xz}] + A_{xxzx}[T_{xz}^T] + A_{xxyz}[T_{yz}] + A_{xxzy}[T_{yz}^T]
\end{aligned} \tag{5.26}
$$

$[T]$ sub-matrices coincide with the linear part of our global stiffness matrix $[K]$. In other words, the matrix $[T_{xx}]$ is obtained by

$$
[T_{xx}] = \begin{bmatrix} q_x^2 & -p_{5968}q_x & -p_{3829}q_x & -p_{2635}q_x \\ -p_{5968}q_x & p_{5968}^2 & p_{5968}p_{3829} & p_{5968}p_{2635} \\ -p_{3829}q_x & p_{5968}p_{3829} & p_{23829} & p_{3829}p_{2635} \\ -p_{2635}q_x & p_{5968}p_{2635} & p_{3829}p_{2635} & p_{2635}^2 \end{bmatrix} \tag{5.27}
$$

following the short notation defined by Pedersen, e.g., $p_{5968} = p_5p_9 - p_6p_8$. When we expand the unknown term, q_x, in Equation 5.27, it becomes $-\beta_1$ in our method:

$$
\begin{aligned}
q_x &= p_{5968} + p_{3829} + p_{2635} \\
q_x &= (y_3z_4 - y_4z_3) + (z_2y_4 - z_4y_2) + (y_2z_3 - y_3z_2) \\
\beta_1 &= -y_3z_4 + y_4z_3 - z_2y_4 + z_4y_2 - y_2z_3 + y_3z_2 \\
q_x &= -\beta_1
\end{aligned} \tag{5.28}
$$

As it is seen from Equation 5.29, the other terms that Pedersen used are the same as the ones used in our method.

$$
\begin{aligned}
q_x &= -\beta_1 \\
p_{5968} &= \beta_2 \\
p_{3829} &= \beta_3 \\
p_{2635} &= \beta_4 \\
q_y &= -\gamma_1 \\
p_{6749} &= \gamma_2 \\
p_{1937} &= \gamma_3 \\
p_{3416} &= \gamma_4 \\
q_z &= \delta_1 \\
p_{4857} &= \delta_2 \\
p_{2718} &= \delta_3 \\
p_{1524} &= \delta_4
\end{aligned}
\tag{5.29}
$$

Apart from the stiffness matrix calculation, the solutions of the nonlinear equations in both methods are the same. Both approaches use the Newton-Raphson method to find the unknown displacements. Hence, the comparison of the computation time required to calculate the stiffness matrices is sufficient to compare the performances of two approaches.

Chapter 6

Experimental Results

We conducted eight experiments to compare the linear and nonlinear finite element methods. Moreover, we compared the proposed nonlinear FEM method with the Pedersen's method [?].

First, we present how we construct FEM models and continue with error analysis for linear and nonlinear FEM solution with the cube mesh. We make analysis with increasing the mesh's density and comparing the displacements for a selected node.

In the first experiment, our aim is to observe the strain-displacement relationship. The test model is a cube with six elements. We also examine the force-displacement relationship for a selected node to compare the displacements for linear and nonlinear FEMs.

The rest of the experiments are performed with different test models. Our aim in these experiments is to compare the accuracy of the deformations for linear and nonlinear FEMs. The results for these experiments are interpreted by comparing displacement amounts for the force applied nodes and all the nodes. Finally, the computational costs of different methods are compared, including experiments on single-core and multi-cores to assess the parallelization of the methods.

6.1 Construction of the FEM Models

The construction of the FEM models consists of three stages:

1. Reading surface meshes. The meshes for the cube, beam and the cross surface models are constructed manually, and the liver mesh is taken from 3D Mesh Research Database [?].

2. Tetrahedralization of the surface mesh using TetGen [?]. We also improve the quality of the models using TetGen.

3. Interactive specification of the constrained (fixed) nodes and the nodes to which the forces to be applied.

6.2 Load Steps

Multiple load steps are used when the load forces are time-dependent or simulation is dynamic [?]. Our simulation is static not time dependent so we used single load step in all of our experiments.

6.3 Material Properties

We used linear material properties for the models in the experiments. We used 1 for Young's modulus (ϵ), and 0.25 for Poisson's ratio (ν).

6.4 Error Analysis

The error analysis is one of the crucial steps of the finite element method to assess the quality of the computed results. We need to make error analysis using approximate results when the exact solution is not available. The error analysis

Element	Displacement - z	Element	Displacement - z	Error (%)
6	0.3831	48	0.3995	4.105
48	0.3995	384	0.4027	0.794
384	0.4027	1536	0.4025	0.049

Table 6.1: Element displacements (in centimeters) along the z-axis for node 4 and their corresponding error ratios for linear FEM

is performed by comparing the displacements of the two approximate results by increasing the number of elements in meshes uniformly. We choose a cube mesh of size 10cm^3 to work with because uniformly increasing the number of elements of the cube is much easier than using a complex mesh.

□

Figure 6.1: The cube mesh with six elements (left) and 48 elements (right).

$$\| u^d - u^{\frac{d}{2}} \| = C \| (u - u^{\frac{d}{2}}) \| \tag{6.1}$$

The error analysis is achieved by comparing the displacements with mesh density d and $\frac{d}{2}$ in 1D (Equation 6.1). If we adapt the 1D formula to 3D, we need to increase the density by 8-times (for every dimension by d to $\frac{d}{2}$) for error analysis. Figure 6.1 shows that number of elements are increased from 6 to 48 for the first step.

The force amount must be the same for each step to observe the displacement errors. Hence, the cube is constrained from the bottom face and pulled towards the direction of the black arrow with same amount of force uniformly distributed among the green nodes (4 units for both the 6- and 48-element meshes) for each step. We choose the node that is highlighted by red arrow to observe the displacements. Moreover, we limited our analysis with 1536 elements because of the high computational cost of nonlinear FEM.

The results in Table 6.1 and 6.2 show that the difference $u^d - u^{\frac{d}{8}}$ decreases with mesh refinement in each step. Using Equation **??**, we can state that the solutions of the linear and nonlinear FEM are valid and converges.

Element	**Displacement - z**	**Element**	**Displacement - z**	**Error (%)**
6	0.3622	48	0.3795	4.558
48	0.3795	384	0.3751	1.159
384	0.375162	1536	0.375101	0.016

Table 6.2: Element displacements (in centimeters) along the z-axis for node 4 and their corresponding error ratios for nonlinear FEM

□

Figure 6.2: Linear FEM error analysis with *L2* and *Energy* norms

The error norms are required to compute the error for the whole solution. *L2* and *Energy* norms are the most frequently used norms to compute the errors. They are defined as

$$L_2 = \sqrt{\int \int \int e^2 \, dx \, dy \, dz} \text{ and}$$
$$Energy = \sqrt{\tfrac{1}{2} \int \int \int \tfrac{\partial e}{\partial x} + \tfrac{\partial e}{\partial y} + \tfrac{\partial e}{\partial z} \, dx \, dy \, dz}, \tag{6.2}$$

where e is the error. The error is computed by subtracting the actual solution u for mesh density d from the approximate solution u_N for mesh density $\frac{d}{2}$. Figures 6.2 and 6.3 show that the error decreases linearly and converges with mesh refinement in each step.

□

Figure 6.3: Nonlinear FEM error analysis with *L2* and *Energy* norms

6.5 Experiment 1

The first experiment is conducted with a cube mesh that has eight nodes and six tetrahedral elements. Figure 6.4 shows that the cube is constrained from the upper four nodes and pulled downwards with a small amount of force (one unit force for each of the upper four nodes). This experiment is conducted with such a small mesh in order to examine the nodal displacements and strains for each element explicitly. Tables 6.3 and 6.4 show force displacements at node 4 using the linear and nonlinear FEMs, respectively. Figures 6.5 and 6.6 show the initial and final positions of the nodes for the linear and nonlinear FEMs, respectively. As it is seen in Figures 6.5 and 6.6, the linear and nonlinear methods produce similar displacements when the force magnitude is small. Table 6.5 gives a comparison of the 1^{st} element strain for the linear and nonlinear FEMs. Table 6.5 shows that even the force magnitude is small, there are differences in strains that can affect displacements. Figure 6.8 shows that the displacement increases linearly with the force magnitude. However, nonlinear FEM behaves exponentially as expected due to the nonlinear strain definitions. Figure 6.7 depicts the convergence of the Newton-Raphson method for the nonlinear FEM.

□

Figure 6.4: Experiment 1: A cube mesh of size 10 cm^3 with eight nodes and six tetrahedra is constrained from the blue nodes and is pulled downwards from the green nodes.

Node	**Displacement - x**	**Displacement - y**	**Displacement - z**
1	0.027234	0.011064	-0.289965
2	0.004306	-0.109719	-0.440739
3	-0.066065	-0.056547	-0.343519
4	-0.107536	0.070143	-0.514524
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0

Table 6.3: Force displacements (in centimeters) at node 4 using linear FEM.

□

Figure 6.5: The initial and final positions of the nodes for the linear FEM. The red spheres show the initial positions and the green spheres show the final positions of the nodes.

Node	Displacement - x	Displacement - y	Displacement - z
1	0.029911	0.012665	-0.278365
2	0.008606	-0.103350	-0.415594
3	-0.058835	-0.051901	-0.324126
4	-0.098945	0.068928	-0.478495
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0

Table 6.4: The displacements (in centimeters) of the nodes using the nonlinear FEM.

□

Figure 6.6: The initial and final positions of the nodes for the nonlinear FEM. The red spheres show the initial positions and the green spheres show the final positions of the nodes.

	Linear FEM	Nonlinear FEM	Error (%)
s_{xx}	0	0	-
s_{yy}	0	0	-
s_{zz}	0.0290	0.0282	-2.76
s_{xy}	0	0	-
s_{xz}	-0.0027	-0.0030	11.11
s_{yz}	-0.0011	-0.0013	18.18

Table 6.5: Comparison of the 1^{st} element strain. The error represents the linear FEM's strain error according to the nonlinear FEM's strain.

$\square$

Figure 6.7: Newton-Raphson convergence graphics for the nonlinear FEM.

$\square$

Figure 6.8: Force displacements (in centimeters) at node 4 for the linear and nonlinear FEMs.

6.6 Experiment 2

The second experiment is conducted with a cube but with 82 nodes and 224-tetrahedral elements. Figure 6.9 shows that the cube is constrained from the bottom face and pulled upwards with a small amount of force (one unit force for each of the lower four nodes). This experiment is conducted with more tetrahedral elements in order to examine the displacement differences and the shape of the mesh after applying two methods.As it is seen from Figures 6.10 and 6.11, the linear and nonlinear methods produce similar displacements because small and large strains provide similar displacements when the force magnitude is small. Therefore, the overall nodal displacement error becomes 0.65%. However, the difference between the results of two methods can be seen from the upper part of the cube; the displacement at the upper face of the cube with linear FEM is more compared to the nonlinear FEM. It can be observed that the force applied nodes (green nodes) produce 5.45% of the error. Moreover, the shape of the cube is more distorted with linear FEM; the left and the right sides of the cube are bent more in linear FEM; the shape of the cube is preserved better with nonlinear FEM.

□

Figure 6.9: Experiment 2: A cube mesh of size 10 cm^3 with 82 nodes and 224-tetrahedra is constrained from blue nodes and is pulled along the arrow from green nodes.

□

Figure 6.10: The final shape of the mesh for the linear FEM (top left: wireframe tetrahedral mesh; top right: wireframe tetrahedral mesh with nodes; bottom left: wireframe surface mesh; bottom right: shaded mesh)

□

Figure 6.11: The final shape of the mesh for the nonlinear FEM (top left: wire-frame tetrahedral mesh; upper right: wireframe tetrahedralmesh with nodes; lower left: wireframe surface mesh; lower right: shaded mesh)

6.7 Experiment 3

The third experiment is conducted with the beam that has 90 nodes and 216-tetrahedral elements. Figures 6.12 (a) and (b) show that the beam is constrained from the blue nodes, and twisted from the both ends of the beam. This experiment is conducted to observe different effects of the nonlinear and the linear FEM deformations on the beam. In the twist experiment, the differences can be seen better. Figures 6.13 and 6.14 show that the nodes that generate the edges of the beam differ (shown with arrow) from each other. In Figure 6.13, the nodes are straight, which is not the desired result of the twist operation. This is the result of usage of linear strains so that linear FEM produced straight displacement. However, at Figure 6.14 nodes are curvy, which is is the expected result of the twist operation. Overall nodal displacement error is 3.10% due to the curvy twist of the nonlinear FEM. It can be observed from the force applied nodes(green nodes), force applied nodes produces 9.61% of error, in linear FEM general shape of the face that hosts the force applied nodes, is more distorted than the nonlinear FEM.

□
(a)

□
(b)

Figure 6.12: Experiment 3: The beam mesh is constrained from the blue nodes and twisted from the green nodes. (a) Front view; (b) Side view, which also shows the force directions applied on each green node.

□

Figure 6.13: Linear FEM solution (top left: wireframe tetrahedra and nodes; top right: only nodes; bottom left: wireframe surface mesh; lower right: shaded mesh).

□

Figure 6.14: Nonlinear FEM solution (top left: wireframe tetrahedra and nodes; top right: only nodes; bottom left: wireframe surface mesh; lower right: shaded mesh).

6.8 Experiment 4

The fourth experiment is conducted with the same beam (Section 6.7) that has 90 nodes and 216-tetrahedral elements. Figure 6.15 shows that the beam is constrained from the blue nodes, and pushed downwards at the green nodes. This experiment is conducted to observe different effects of the nonlinear and the linear FEM deformations over the beam mesh. It can be observed that with linear FEM, the beam is bent more than with nonlinear FEM. The width of the beam become wider with the linear FEM at the both ends (see Figure 6.16). On the other hand, the deformation is smoother with nonlinear FEM (see Figure 6.17). This volume difference results in overall 4.32% error among all nodes, and 15.95% error on force applied nodes (green nodes).

□

Figure 6.15: Experiment 4: The beam mesh is constrained from the blue nodes and pushed downwards at the green nodes.

□

Figure 6.16: Linear FEM solution (top: wireframe tetrahedra and nodes; middle upper: shaded mesh; middle lower: initial mesh and the final tetrahedra are overlaid; bottom: initial and final meshes are overlaid).

□

Figure 6.17: Nonlinear FEM solution (top: wireframe tetrahedra and nodes; middle upper: shaded mesh; middle lower: initial mesh and the final tetrahedra are overlaid; bottom: initial and final meshes are overlaid).

6.9 Experiment 5

This experiment is conducted with the cross mesh that has 159 nodes and 244-tetrahedral elements. Figure 6.18 shows that the cross-shape is constrained from the blue nodes and pushed towards the green nodes. This experiment is conducted to observe the different effects of nonlinear and linear FEM deformations over the cross-shaped mesh with high amount of force (50 units). It can be observed that under a high amount of force, linear FEM produces unexpected result by expanding the upper and the lower part of the mesh (cf. Figures 6.19 and 6.20). As a result of that, the overall nodal displacement and force node displacement errors are 213.36% and 232.56%, respectively. It can be said that under a high amount of force, nonlinear FEM produces accurate, thus more realistic, results (cf. Figures 6.21 and 6.22).

□

Figure 6.18: Experiment 5: The cross mesh is constrained from the blue nodes and pushed towards the green nodes.

□
(a)

□
(b)

Figure 6.19: Linear FEM solution: (a) wireframe mesh; (b) shaded mesh.

□
(a)

□
(b)

Figure 6.20: Linear FEM solution: (a) initial and final wireframe meshes are overlaid; (b) initial and final shaded meshes are overlaid.

□
(a)

□
(b)

Figure 6.21: Nonlinear FEM solution: (a) wireframe mesh; (b) shaded mesh.

□
(a)

□
(b)

Figure 6.22: Nonlinear FEM solution: (a) initial and final wireframe meshes are overlaid; (b) initial and final shaded meshes are overlaid.

6.10 Experiment 6

The sixth experiment is conducted with the liver mesh [?] that has 465 nodes and 1560-tetrahedral elements. Figure 6.23 shows that the liver mesh is constrained from the blue nodes, and pulled from the green nodes towards the arrow direction. This experiment is conducted to observe the different effects of the nonlinear and the linear FEM deformations over the liver. Linear FEM produces a protrusion at the top of the mesh (see Figure 6.24). It can be observed that the liver mesh is deformed more realistically and smoothly with nonlinear FEM (see Figure 6.25). As a result of the protrusion generated for the linear FEM, the node displacement error becomes 12.85%. Apart from the force-applied region, the overall shape is preserved (the overall nodal displacement error is 0.72%) in both methods due to the low amount of force. We can conclude that with dense meshes, nonlinear FEM produces accurate, thus more realistic, results.

□

Figure 6.23: Experiment 6: The liver mesh is constrained from the blue nodes and pulled from the green nodes (left: initial nodes; right: initial shaded mesh and nodes).

□
(a)

□
(b)

□
(c)

Figure 6.24: Linear FEM solution: (a) left: nodes, right: tetrahedral wireframe mesh; (b) left: wireframe surface mesh, right: wireframe surface mesh with nodes; (c) left: shaded mesh, right: shaded mesh with nodes.

□
(a)

□
(b)

□
(c)

Figure 6.25: Nonlinear FEM solution: (a) left: nodes, right: tetrahedral wireframe mesh; (b) left: wireframe surface mesh, right: wireframe surface mesh with nodes; (c) left: shaded mesh, right: shaded mesh with nodes.

6.11 Experiment 7

This experiment is conducted with the liver mesh that has 465 nodes and 1560-tetrahedral elements. Figure 6.26 shows that the liver mesh is constrained from the blue nodes, and pulled from the green node towards the arrow direction. This experiment is conducted to observe the different effects of the nonlinear and the linear FEM deformations over the liver with pulling only one node. It can be observed that the linear FEM produces a high amount of displacement around the force node (see Figure 6.27). As a result of that, the node displacement error becomes 58.94%. The liver mesh is deformed more realistically and smoothly with nonlinear FEM (see Figure 6.28). Apart from the force-applied region, the overall shape is preserved (the overall nodal displacement error is 0.12%). It can be said that with dense meshes, nonlinear FEM produces accurate, thus more realistic, results.

□

Figure 6.26: Experiment 7: The liver mesh is constrained from the blue nodes and pulled from the green node (left: initial nodes, right: initial shaded mesh and nodes).

□
(a)

□
(b)

□
(c)

Figure 6.27: Linear FEM solution: (a) left: nodes, right: tetrahedral wireframe mesh; (b) left: wireframe surface mesh, right: wireframe surface mesh with nodes; (c) left: shaded mesh, right: shaded mesh with nodes.

□
(a)

□
(b)

□
(c)

Figure 6.28: Nonlinear FEM solution: (a) left: nodes, right: tetrahedral wireframe mesh; (b) left: wireframe surface mesh, right: wireframe surface mesh with nodes; (c) left: shaded mesh, right: shaded mesh with nodes.

6.12 Experiment 8

This experiment is again conducted with the liver mesh that has 465 nodes and 1560-tetrahedral elements. Figure 6.29 shows that the liver mesh is constrained from the blue nodes, and pushed upwards at the green nodes. This experiment is conducted to observe the different effects of the nonlinear and the linear FEM deformations over the liver when pushing the liver from several nodes. Linear FEM produced high amount of displacement around the force node (see Figure 6.30). As a result of that, the node displacement error becomes 17.28%. It can be observed that the liver mesh is deformed more realistically and smoothly with nonlinear FEM (see Figure 6.31). The mesh is more collapsed inwards with linear FEM, whereas its structure is better preserved with nonlinear FEM. Apart from the force applied region, the overall shape is preserved (the overall nodal displacement error is 0.1%).

□

Figure 6.29: Experiment 8: The liver mesh is constrained from the blue nodes and pushed towards the green nodes (left - initial nodes, right - initial shaded mesh and nodes).

□
(a)

□
(b)

Figure 6.30: Linear FEM solution: (a) left: nodes, right: shaded mesh with nodes; (b) the mesh from a different view, left: shaded mesh with nodes, right: shaded mesh.

□
(a)

□
(b)

Figure 6.31: Nonlinear FEM solution: (a) left: nodes, right: shaded mesh with nodes; (b) the mesh from a different view, left: shaded mesh with nodes, right: shaded mesh.

6.13 Computational Cost Analysis

The computation times of the finite element experiments are required to make comparison of how much our proposed solution is faster than Pedersen's solution. Moreover, we can observe that nonlinear FEM has higher computation cost than linear FEM. However, high computation cost gives us much more accurate results that we can ignore this high cost when we are working with crucial simulations like car crash tests, surgical simulators (in terms of accuracy) and concrete analysis of the building.

When comparing nonlinear solutions, we calculated the computation times of construction of the stiffness matrices and the whole solution in order to state how different calculation of stiffness matrices directly affects the stiffness matrices' and the whole solution's computation time. Moreover, we conducted these experiments on two different systems to analyze how clock speed of the processor affects the computation time and to state the multi-core efficiencies on different systems. We conducted all the experiments on a desktop computer with Core i7 processor overclocked at $4.0GHz$ with 24GB of RAM. Table 6.6, Figures 6.32 and 6.36 show the computation times required to calculate the stiffness matrix. Table 6.7, Figures 6.34 and 6.38 show the computation times required to solve the system.

Figures 6.33, 6.35, 6.37, 6.39 and 6.40 depict the speed-ups of each experiment obtained by using the proposed approach with respect to the Pedersen's method for different number of cores and threads. The speed-up is calculated by

$$\text{Speed-up} = \frac{\text{Runtime(Pedersen's method)}}{\text{Runtime(The proposed approach)}} \tag{6.3}$$

Figure 6.41 shows the speed-up of multi-core over the single-core on our system. Multicore efficiency obtained using a multi-core with respect to a single core is given by

$$\text{Multicore Efficiency} = \left(\frac{\text{Runtime(Single-core)}}{\text{Runtime(Multi-core)}}\right) \times 100 \tag{6.4}$$

We used Matlab's Parallel Computing Toolbox to implement multithreading. The toolbox provides local workers (Matlab computational engines) that distributes the program into threads to execute applications on a multicore system [**?**]. We implemented multithreading by using $parfor$ loop instead of for loop. When iterating over the elements to calculate stiffness matrices, residuals and tangent stiffness matrices, part of the computation is stayed on main Matlab worker, and the rest of the parts are computed on local workers. When $parfor$ loop starts, necessary data is sent from the main thread to local workers, and at the end of the $parfor$ loop, the results are sent back to the main thread and combined together.

The proposed solution is highly parallelizable; our program works 3.6 times faster when it is distributed on 4-core. Overhead of creating local workers is relatively high when number of elements is small. In this case, multi threaded solution becomes less efficient than the single threaded solution (cf. Figure 6.41).

The proposed method outperforms the Pedersen's method. On the average, it is 111% faster at computing stiffness matrices since Pedersen's method uses much more symbolic terms (cf. Figure 6.40). However, both methods uses Newton-Raphson method to solve nonlinear equations which takes 90% of the computation time. Therefore overall speed-up decreases to 16% on average (cf. Figure 6.40).

□

Figure 6.32: Comparison of the computation times required to calculate the stiffness matrix (single thread).

□

Figure 6.33: Relative performance comparison of the stiffness matrix calculation (single thread).

□

Figure 6.34: Comparison of the computation times required to solve the system (single thread).

Experiment	**Linear**	**Pedersen**	**Pedersen MT**	**Prop Non**	**Prop Non MT**
1^{st} - 6	0.0572	0.8341	1.8073	0.3783	0.6109
2^{nd} - 224	0.1753	38.3624	8.4493	19.8544	3.8807
3^{rd} - 226	0.1692	32.5297	8.0717	15.3335	3.7179
4^{th} - 216	0.1699	34.1291	8.0912	16.2847	3.7356
5^{th} - 244	0.1861	36.9055	8.5068	18.2106	4.0998
6^{th} - 1560	0.9178	266.3396	65.1570	125.6929	36.7951
7^{th} - 1560	0.8851	265.0623	65.9921	124.0710	37.0265
8^{th} - 1560	0.9979	266.4710	65.4512	124.4716	37.1006

Table 6.6: Computation times (in seconds) of the stiffness matrices for all experiments (MT: Multi thread, Prop Non: The proposed nonlinear solution).

□

Figure 6.35: Relative performance comparison of the system solution (single thread).

Experiment	**Linear**	**Pedersen**	**Pedersen MT**	**Prop Non**	**Prop Non MT**
1^{st} - 6	0.0592	4.8396	3.4315	4.3774	2.1489
2^{nd} - 224	0.2359	402.0249	94.9423	341.0334	79.4478
3^{rd} - 226	0.2274	460.6526	128.5903	394.5419	103.0223
4^{th} - 216	0.2282	459.8582	125.2817	448.9026	122.0369
5^{th} - 244	0.2557	2267.3736	656.7743	1967.2491	513.2731
6^{th} - 1560	1.4029	4492.5631	909.9521	3574.6976	865.8751
7^{th} - 1560	1.3644	5736.3210	1274.7842	4636.2668	1224.7844
8^{th} - 1560	1.3878	4849.1274	1004.8713	3926.7512	972.4567

Table 6.7: Computation times (in seconds) of the systems for all experiments. (MT: Multi thread, Prop Non: The proposed nonlinear solution).

□

Figure 6.36: Comparison of the computation times required to calculate the stiffness matrix (eight threads on four cores).

□

Figure 6.37: Relative performance comparison of the stiffness matrix calculation (eight threads on four cores).

Figure 6.38: Comparison of the computation times required to solve the system (eight threads on four cores).

Figure 6.39: Relative performance comparison of the system solution (eight threads on four cores).

Figure 6.40: Relative performance comparison averaged over all experiments.

Figure 6.41: Multi-core efficiency of the proposed approach (speed-up of eight threads on four cores over the single core).

Chapter 7

Conclusion and Future Work

We have presented a new non-linear FEM solution method. The proposed solution is easier to analyze in terms of constructing the elemental stiffness matrices and faster than Pedersen's solution. The proposed solution is approximately twice faster on the average at computing stiffness matrices and 17% faster at computing the whole system than the Pedersen's solution.

We compared our solution with linear FEM to see advantages and drawbacks in eight different experiments. Our proposed solution has huge advantages over the linear FEM in terms of accuracy. The proposed solution handles large deformations and small deformations perfectly although difference in small deformations is low. However, this low amount of difference cannot be neglected for applications that require very high accuracy. Parallelization is also important to speed-up the FEM solution. We obtain significant speed-ups on multicore machines.

Although the proposed solution has significant advantages over linear FEM and recent non-linear solution, there is still room for development. Possible future extensions are as follows:

1. Although Newton-Raphson is a fast solution technique, over 90% of the computation time of the whole system spent in Newton-Raphson solution

procedure. It can be implemented better to overcome jumping to the unexpected roots or different solution procedure can be implemented.

2. The proposed solution is highly parallelizable so it can benefit from a GPU implementation. However, the nonlinear solution procedure uses over 6GB of system memory when computing the solution for over 1500 elements, so we need GPUs that has lots of memory.

3. Although we decreased the system memory usage by simplifying the solution procedure for the nonlinear solution, it uses a significant amount of system memory. Hence, the solution procedure can be optimized more to decrease the memory usage.

4. All experiments are conducted with the same material properties. They can be extended by measuring the exact properties of the real objects (i.e., an actual liver).

Bibliography

[1] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] K.-M. G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette across time part ii: Applications to human modeling and markerless motion tracking. *Int. J. Comput. Vision*, 63(3):225–245, July 2005.

[3] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[4] T. Knot. Ogre. open source 3d rendering engine. http://www.ogre3d.org/, 2012.

[5] J. Moan. Documentation architecture. ogre- object oriented rendering engine. http://www.ogre3d.org/tikiwiki/tiki-index.php?page=Documentation+Architecture/, 2012.

[6] OpenNI. Programmer guide-openni. http://openni.org/Documentation/ProgrammerGuide.htm 2012.

[7] PrimeSense. Nite. primesense natural interaction. http://www.primesense.com/en/nite, 2012.

[8] M. Tang. Recognizing hand gestures with microsofts kinect. Technical Report MSU-CSE-00-2, Department of Computer Science, Stanford University, January 2011.

Appendix A

Rhinoplasty Application

We conducted two different experiments to apply our solution in the area of rhinoplasty. In the experiments, we correct the form of misshapen noses (see Figure A.1). We compare the accuracy of the deformations for linear and nonlinear FEMs. The results for these experiments are interpreted by comparing displacement amounts for the force applied nodes and all the nodes.

A.1 Experiment 1

The first experiment is conducted with a head mesh that has 6709 nodes and 25722 tetrahedral elements (see Figure A.2 (a)). Number of tetrahedral elements are very high. However, all the operations are done in the nose area with 1458 tetrahedral elements. To simplify the calculations, stationary tetrahedral elements are not taken into account. Figure A.2 (b) shows that the head mesh is constrained from the blue nodes, and is pushed upwards at the green nodes. This experiment is conducted to observe the different effects of the nonlinear and the linear FEM deformations over the nose. Linear FEM produced high amount of displacement at the upper part of the nose (see Figure A.2 (c)). As a result of that, the node displacement error becomes 64.92%. It can be observed that the

Figure A.1: The perfect nose.

nose was deformed more realistically and smoothly with nonlinear FEM (see Figure A.2 (d)). The nose is more collapsed inwards with linear FEM, whereas its structure is better preserved with nonlinear FEM and the overall shape is more similar to a perfect nose than linear FEM. Although, nearly 6000 nodes are constrained, the overall nodal displacement error is 3.88%.

(a) (b)

(c) (d)

Figure A.2: Experiment 1: (a) Initial misshapen nose. (b) Head mesh is constrained from the blue nodes, and is pushed upwards at the green nodes. (c) Linear FEM Solution: left: wireframe surface mesh with nodes, right: shaded mesh with texture. (d) Nonlinear FEM Solution: left: wireframe surface mesh with nodes, right: shaded mesh with texture.

A.2 Experiment 2

The second experiment is conducted with mesh similar to the one used in the first experiment. However, it has 7071 nodes and 27020 tetrahedral elements due to different shape of the nose and tetrahedralization (see Figure A.3 (a)). To simplify the calculations, stationary tetrahedral elements are not taken into account. 4511 tetrahedra are included in the calculations. A.3 (b) shows that the head mesh is constrained from the blue nodes, and is pushed upwards at the green nodes. This experiment is conducted to observe the different effects of the nonlinear and the linear FEM deformations over the nose. Linear FEM produced high amount of displacement at the lower part of the nose (see A.3 (c)). Moreover, the nose is nearly collapsed inwards that is far away from the perfect nose. As a result of that, the node displacement error becomes 96.03%. With nonlinear FEM, the nose is deformed more realistically and smoothly (see A.3 (d)). The nose is more collapsed inwards with linear FEM, whereas its structure is better preserved with nonlinear FEM and the overall shape is more similar to a perfect nose than linear FEM. Although, nearly 5000 nodes are constrained, the overall nodal displacement error is 11.19%.

(a) (b)

(c) (d)

Figure A.3: Experiment 2: (a) Initial misshapen nose. (b) Head mesh is constrained from the blue nodes, and is pushed upwards at the green nodes. (c) Linear FEM Solution: left: wireframe surface mesh with nodes, right: shaded mesh with texture. (d) Nonlinear FEM Solution: left: wireframe surface mesh with nodes, right: shaded mesh with texture.