



*People's Democratic Republic of Algeria  
Ministry of Higher Education and Scientific Research  
Larbi Tébessi University - Tebessa  
Faculty of Science and Technology  
Department: Mechanical Engineering*

*Final dissertation for obtaining the MASTER degree  
Faculty: Science and Technology  
Department: Mechanical Engineering  
Specialty: Mechanical Construction*

*Theme*

# **Programming and validation of 1D finite elements for bar and beam structures**

*Theme Submitted by:*

*BOUACHMA Mohamed Amine*

*Before the jury:*

<i>President</i>	<i>Dr. HADJAB Abdelhakim</i>	<i>MAA</i>
<i>Supervisor</i>	<i>Pr. DEGHBODJ Samir</i>	<i>Professor</i>
<i>Examiner</i>	<i>Dr. LEMITA Nour El Houda</i>	<i>MAB</i>

*Date of defense: 26/06/2024*

## ملخص

في مجال الإنشاءات الميكانيكية يقوم مهندسو الإنشاءات الميكانيكية بحل عدد من الأنظمة لمختلف التطبيقات، وتميل هذه الأنظمة إلى أن تكون معقدة وتستغرق وقتاً طويلاً وغالباً ما يتم تنفيذها يدوياً. يجب معالجة هذه المشكلة، لذا فإن العملية الآلية لطرق الدراسة هي الحل الأكثر منطقية وقابلية للتحقيق. تُعد منهجية العناصر المحدودة منهجية مهمة، فهي تحل المشاكل الميكانيكية من خلال تقسيم الهيكل إلى أجزاء أصغر، وبالتالي معالجة المشاكل بكفاءة وسرعة عالية. وقد طبقنا هذه المنهجية على ثلاثة أنظمة ميكانيكية هي "bar" و"truss" و"beam". يساهم بحثنا في مجال الإنشاءات الميكانيكية من خلال توفير برنامج يستخدم خوارزميات طريقة العناصر المحدودة كقاعدة لحساب النتائج المطلوبة. ينقسم البرنامج إلى ثلاثة برامج فرعية لأنظمة "bar" و"truss" و"beam". ومن خلال تخفيف الوقت اللازم لحساب أنظمة الدراسة والجهود الشخصية المبذولة في هذا المجال، فإن نهجنا يسرّع هذه العملية مع ضمان نتائج عالية الدقة.

**الكلمات المفتاحية:** طريقة العناصر المحدودة، RDM6، برنامج حسابي، عنصر محدد bar، عنصر محدد truss، عنصر

محدد beam

## *Abstract*

In mechanical construction engineers solve number of systems for various applications, these systems tend to be complex, time consuming and often done manually. This problem needs to be addressed so an automated process of the study methods is the most logical and achievable solution. Finite element method is an important methodology; it solves mechanical problems by devising the structure into smaller segments, therefore treating the problems with high efficiency and speed. We applied this methodology on three mechanical systems which are “Bar”, “Truss” and “Beam”. Our research contributes to the field of mechanical construction by providing a program that uses finite element method algorithms as a base to calculate the required results. The program is separated into three sub-programs for “Bar”, “Truss” and “Beam” systems. By mitigating the time required for study systems calculation and personal efforts, our approach speeds up that process while ensuring high accuracy results.

**Key words:** Mechanical Construction, Finite Element Method, Algorithms, Bar, Truss, Beam.

## *Acknowledgment*

In the name of Allah, our Lord and Creator, Who has blessed us with the gift of reasoning and the pursuit of knowledge, we begin by expressing our gratitude.

I would like to extend my heartfelt appreciation to my supervisors, Pr. DEGHBODJ Samir for his unwavering trust and patience. His insightful guidance, advice, and commitment were instrumental in shaping this dissertation.

I also wish to express my deep gratitude to the esteemed members of the jury Dr. HADJAB Abdelhakim and Dr. LEMITA Nour El Houda agreeing to evaluate and judge this humble work.

I am indebted to **CHERGUI Kheir Eddine**, my brother, who was nothing but supportive, helpful and I always look at you as a perfect role model.

Last but not least, I must express my profound gratitude to my father, my mother, my sisters and my friends. Their unwavering support, continuous encouragement, and countless sacrifices throughout our academic journey made this accomplishment possible.

## *Dedication*

*“To my beloved parents, your encouragement is the*

*Only thing keeps me going.*

*To my sisters and my brother*

*Thanks for always being there for me.*

*To all my friends, the ones near my heart,*

*Thank you all; I dedicate this work to you”*

# *List of Content*

<i>ملخص</i> .....	I
<i>Abstract</i> .....	II
<i>Acknowledgment</i> .....	III
<i>Dedication</i> .....	IV
<i>List of Content</i> .....	V
<i>List of Abbreviations</i> .....	IX
<i>List of Tables</i> .....	X
<i>List of Figures</i> .....	XII
<i>General Introduction</i> .....	1
<i>Chapter 1 Bibliographic research and state of the art</i> .....	3
1.1 Introduction .....	3
1.2 Element finite method .....	3
1.2.1 Historical background .....	3
1.2.2 Definition .....	4
1.3 Steps of the Finite Element Method .....	5
1.4 The benefits of Finite Element Method.....	6
1.5 Truss systems.....	7
1.5.1 Definition .....	7
1.5.2 The principle of a truss system.....	8
1.5.3 Advantages of truss system .....	8
1.6 Beam element .....	8
1.6.1 Introduction .....	8
1.6.2 Definition of beams .....	9
1.6.3 Types of beams.....	9

1.6.4	Flexural behavior of beams .....	11
1.6.5	Euler-Bernoulli Beam Theory .....	11
1.7	Conclusion .....	11
<i>Chapter 2 Finite element one dimensional 1D .....</i>		<b>13</b>
2.1	Introduction .....	13
2.2	Fundamental concepts .....	13
2.2.1	Domain discretization .....	13
2.2.2	Shape functions .....	13
2.2.3	Stiffness matrix .....	13
2.3	Solution procedure.....	14
2.3.1	Establishing equilibrium equations .....	14
2.3.2	Assembling the equation System .....	14
2.3.3	Applying boundary conditions .....	14
2.4	Bar element.....	14
2.4.1	Governing equations .....	14
2.4.2	Equilibrium equations .....	14
2.4.3	Approximate solution using the finite element method .....	16
2.4.4	Example of the calculation for a bar element in tension .....	19
2.5	Truss element.....	23
2.5.1	Definition of truss systems .....	23
2.5.2	Characteristics of truss systems.....	24
2.5.3	Formulation of the truss finite element .....	24
2.5.4	Stress State of a Truss Finite Element.....	27
2.5.5	Illustration demonstrating the computation for a truss finite element .....	28
2.6	Beam elements.....	33
2.6.1	Definition of beam systems.....	33
2.6.2	Beam elements .....	33
2.6.3	Bending behavior of beam bar finite element .....	34
2.6.4	Formulation beam element.....	36
2.6.5	Example calculation of beam element .....	42
2.7	Conclusion .....	47

<i>Chapter 3 Programming and validation</i> .....	48
3.1 Programming Languages .....	48
3.1.1 Overview of programming languages .....	48
3.1.2 Core Concepts in Programming .....	48
3.1.3 Development Paradigms .....	49
3.1.4 Development Tools .....	49
3.1.5 Software Development Lifecycle.....	49
3.1.6 Popular Programming Languages and Their Uses .....	50
3.1.7 Emerging Trends .....	50
3.2 Chosen Programming Language (Python) .....	51
3.2.1 Definition and history of python .....	51
3.2.2 Python capabilities and functions.....	52
3.2.3 Errors and exceptions .....	52
3.2.4 Jupyter notebook .....	54
3.2.5 Overview of Jupyter notebook .....	54
3.2.6 Key features of Jupyter notebook.....	55
3.3 Finite element program.....	55
3.3.1 Finite element program exaction steps.....	55
3.3.2 Developed program .....	56
3.3.3 Code .....	56
3.3.4 Bar program.....	56
3.3.5 Truss program .....	60
3.3.6 Beam program: .....	61
3.4 Validating results using RMD6 software .....	63
3.4.1 Overview of RDM6.....	63
3.4.2 RDM6 Features .....	64
3.4.3 RDM6 Modules.....	64
3.5 Comparing program results with traditional analytical results.....	64
3.5.1 Example of the calculation for bar element in tension.....	65
3.5.2 Example of the calculation for truss element .....	66
3.5.3 Example of the calculation for a bar element in bending.....	67



3.6 Comparing program results with RDM6 results in complex cases .....68

    3.6.1 Illustration of the computation for a truss element .....68

    3.6.2 Illustration of the computation for a beam element .....73

3.7 Conclusion.....78

*General Conclusion*..... 79

*References* ..... 80

## *List of Abbreviations*

1D	One dimensional
2D	Two dimensional
3D	Three dimensional
BSD	Berkeley Software Distribution
DOF	Degrees of freedom
FEA	Finite Elements Analysis
FEM	Finite Elements Method
Git	Global information tracker
HVAC	heating, ventilation, and air conditioning
IDEs	Integrated Development Environments
IoT	Internet of Things
OOP	object-oriented programming
OS	operating system
RDM 6	Résistance de matériaux 6
SVN	Subversion

## *List of Tables*

<b>Table 2.1:</b> The boundary conditions specific to the case study (bar in tension) .....	22
<b>Table 2.2:</b> Results specific to the case study (bar in tension).....	23
<b>Table 2.3:</b> The structure’s geometry characteristics (truss structure under tensile loading).....	29
<b>Table 2.4:</b> The boundary conditions specific to the case study (truss structure under tensile loading).....	32
<b>Table 2.5:</b> Outcomes specific to the case study (truss structure under tensile loading).....	33
<b>Table 2.6:</b> The boundary conditions specific to the case study (beam under bending) .....	45
<b>Table 2.7:</b> Results specific to the case study (beam under bending).....	47
<b>Table 3.1:</b> Some standard Python built-in exceptions.....	54
<b>Table 3.2:</b> The main modules available in the RDM6 .....	64
<b>Table 3.3:</b> Outcomes provided by manual computing for bar element .....	65
<b>Table 3.4:</b> Results derived from proposed program for bar element.....	65
<b>Table 3.5:</b> Relative error between outcomes provided by manual computing and those from proposed program for bar element .....	66
<b>Table 3.6:</b> Results obtained through manual calculations for truss element .....	66
<b>Table 3.7:</b> Results derived from the proposed program for truss element .....	66
<b>Table 3.8:</b> Relative error between outcomes provided by manual computing and those from proposed program for truss element.....	67
<b>Table 3.9:</b> Results derived from manual calculations for beam element .....	67
<b>Table 3.10:</b> Results derived from the proposed program for beam element .....	67
<b>Table 3.11:</b> Relative error between the outcomes obtained through manual computation and those generated by the proposed program for the beam element.....	68
<b>Table 3.12:</b> Results provided by the developed calculation program for the truss structure consisting of 7 bars.....	69
<b>Table 3.13:</b> Outcomes provided by RDM6 software for the truss structure consisting of 7 bars ..	73
<b>Table 3.14:</b> Relative error between the results generated by the RDM6 software and those produced by the proposed program for the truss structure consisting of 7 bars.....	73
<b>Table 3.15:</b> Results provided by the developed calculation program for the beam structure .....	74
<b>Table 3.16:</b> Outcomes provided by RDM6 software for the beam structure .....	77

**Table 3.17:** Relative error between the results generated by the RDM6 software and those produced by the proposed program for the beam structure.....77

## *List of Figures*

<b>Figure 1.1:</b> Examples of 1D finite element: a) Bar element; b) Bar element subjected to a distributed load with density $q(x)$ ; c) Beam element [5]; d) Beam element subjected to distributed load and concentrated force [5].....	4
<b>Figure 1.2:</b> Examples of 2D finite elements: a) Triangular element with 3 nodes; b) Plate modeled by 2 triangular elements with 3 nodes; c) Quadrilateral element with 8 nodes; d) Meshing of a plate with quadrilateral elements.....	5
<b>Figure 1.3:</b> Flowchart of the various steps of the finite element method.....	6
<b>Figure 1.4:</b> Truss system in bridges .....	7
<b>Figure 1.5:</b> Trussing buildings .....	7
<b>Figure 1.6:</b> Example of application of truss system in the manufacturing of race cars .....	8
<b>Figure 1.7:</b> Schematic representation of a beam .....	9
<b>Figure 1.8:</b> Schematic representation of a simple beam .....	9
<b>Figure 1.9:</b> Schematic representation of cantilever beam .....	10
<b>Figure 1.10:</b> Example of application of cantilever beam .....	10
<b>Figure 1.11:</b> Schematic representation of composite .....	10
<b>Figure 1.12:</b> Composite beams used in bridge construction .....	10
<b>Figure 1.13:</b> Schematic representation of truss beam .....	11
<b>Figure 2.1:</b> Schematic representation of a bar, subjected to a tensile load $q$ .....	15
<b>Figure 2.2:</b> Equilibrium of forces for an elemental segment of the bar with length $dx$ .....	16
<b>Figure 2.3:</b> Schematic representation of a two-node linear bar finite element .....	17
<b>Figure 2.5:</b> Discretization of the beam into 3 linear elements and 4 nodes .....	20
<b>Figure 2.4:</b> Example of the calculation of a beam subjected to tensile force.....	20
<b>Figure 2.6:</b> Schematic representation of a truss system with triangular elements (Pratt truss bridge) .....	24
<b>Figure 2.7:</b> Schematic representation of truss element with two nodes (i) and (j), inclined at an angle $\theta$ .....	25
<b>Figure 2.8:</b> Example of the calculation of a truss structure under tensile loading.....	29
<b>Figure 2.9:</b> Schematic representation of a beam system.....	33
<b>Figure 2.10:</b> Beam system under bending load $q(x)$ .....	35

<b>Figure 2.11:</b> Representative beam element showing displacements and rotations due to bending load .....	36
<b>Figure 2.12:</b> Schematic representation of the degrees of freedom of a beam bar element .....	37
<b>Figure 2.13:</b> Transformation of the distributed load into equivalent nodal loads .....	42
<b>Figure 2.14:</b> Illustrates a demonstration of calculating the response .....	43
<b>Figure 2.15:</b> Discretization of the beam into 2 linear elements and 3 nodes .....	43
<b>Figure 3.1:</b> popular programming languages .....	48
<b>Figure 3.2:</b> Software Development Lifecycle .....	50
<b>Figure 3.3:</b> python logo .....	51
<b>Figure 3.4:</b> Undefined variable error example in python .....	53
<b>Figure 3.5:</b> Jupyter Logo .....	54
<b>Figure 3.6:</b> program execution steps .....	56
<b>Figure 3.7:</b> Used python libraires .....	56
<b>Figure 3.8:</b> Bar nodes parameters .....	57
<b>Figure 3.9:</b> Bar element parameters .....	57
<b>Figure 3.10:</b> Bar elementary stiffness matrices calculation .....	58
<b>Figure 3.11:</b> Bar global matrix calculation .....	58
<b>Figure 3.12:</b> Bar Gaussian elimination process .....	59
<b>Figure 3.13:</b> Bar results .....	59
<b>Figure 3.14:</b> Truss node parameters .....	60
<b>Figure 3.15:</b> Truss elementary stiffness matrices calculation .....	60
<b>Figure 3.16:</b> Truss Gaussian elimination process .....	61
<b>Figure 3.17:</b> Truss results .....	61
<b>Figure 3.18:</b> Beam node parameters .....	62
<b>Figure 3.19:</b> Beam element parameters .....	62
<b>Figure 3.20:</b> Beam elementary stiffness matrices calculation .....	62
<b>Figure 3.21:</b> Beam Gaussian elimination process .....	63
<b>Figure 3.22:</b> Beam results .....	63
<b>Figure 3.23:</b> A truss structure consisting of 7 bars and 5 nodes .....	69
<b>Figure 3.24:</b> Creating the truss model consisting of 7 bars and 5 nodes .....	70
<b>Figure 3.25:</b> Properties of the material used .....	70

**Figure 3.26:** Dimensions and shape of the chosen section of the bar .....71

**Figure 3.27:** Applying loads and boundary conditions .....71

**Figure 3.28:** Viewing the Results.....72

**Figure 3.29:** Exporting the report results .....72

**Figure 3.30:** A beam structure consisting 5 nodes .....74

**Figure 3.31:** Creating the beam model consisting 5 nodes .....74

**Figure 3.32:** Properties of the material used .....75

**Figure 3.33:** Dimensions and shape of the chosen section of the beam.....75

**Figure 3.34:** Applying loads and boundary conditions .....76

**Figure 3.35:** Viewing the Results.....76

**Figure 3.36:** Exporting the report results .....77

## *General Introduction*

Mechanical construction is a key element in making mechanical systems; it includes building, installing and maintaining processes of various types of structures, where the engineer tracks heating, ventilation, and air conditioning (HVAC) and often other special phenomenon. These systems must follow 5 phases; Design and planning, Fabrication, Installation, Commissioning and Maintenance and repair.

Our work is concentrated in the first phase just before fabrication, studying 3 mechanical systems which are “Bar”, “Truss” and “Beam”. The systems are widely used not only in the mechanical field, but other fields such as civil engineering which further amplifies its importance.

Of the 3 mechanical systems mentioned before we start with the Bar system; the easiest system and the most used one, it is processed in one axis. The other two are processed in two axes, the Truss system is simply a number of bar systems linked together in a geometrical way. We see it most on bridges. Beam system in the other hand treats different factors like Flexion and bending.

To study the mechanical systems a method must be used, in our case we are using the Finite Element Method, its implementation as a methodology in structural and computational mechanics is now an essential component of engineering practice. The complexity of modern structures and mechanical components has led to a growing demand for sophisticated computational tools that can accurately predict their behavior under a variety of conditions. The objective of this dissertation is to present a comprehensive programming methodology and validation procedure for multi-dimensional finite elements. Finite Element Method solves this problem mainly by enabling the resolution of complex problems by decomposing a structure into smaller, more manageable elements.

In the background to our study, we aimed to automate the finite element method process and make it usable by the average person. To achieve that we developed a program written in python where the program asks the user for specific variables and returns the results.

On this dissertation we present 3 main chapters;



The first contains a review of literature on Finite Element Method as well as its Phases and how important it is in Mechanical Construction, then concluding by identifying the gaps and areas that require further investigation.

Chapter 2 goes deep into Finite Element Method, talking about its concepts and formulas, then specifically mention the 3 targeted systems, Bar, Truss and Beam; explaining what are they and how are they formulated, as well as mentioning their key features and general applications.

The Last Chapter talks about programming and what programming language we used. Then presenting our program including step by step explanation and the main differences between the three systems, last but not least, we validated our program's results with the RDM6 software as well as traditional analytic results (hand calculated).

In the conclusion we will summarize the main findings. Contributions of the dissertation to mechanical construction and limitations of the dissertation and suggestions for future research

# Chapter 1

## *Bibliographic research and state of the art*

### *1.1 Introduction*

Nonlinear partial differential equations are of interest to mathematicians, physicists and engineers for their rich mathematical structure and properties, and for their applications in fluid mechanics, plasma physics, fiber optics, condensed matter physics and chemistry. Finding how to solve them is important. To understand the properties of the solutions, these nonlinear partial differential equations can be solved exactly or numerically [1]. Recently, engineering issues have become less complicated, and this is due to the finite element method. It has become one of the most famous and widely used methods. It is a powerful tool for solving numerical problems and can be used in almost all fields [2].

### *1.2 Element finite method*

#### *1.2.1 Historical background*

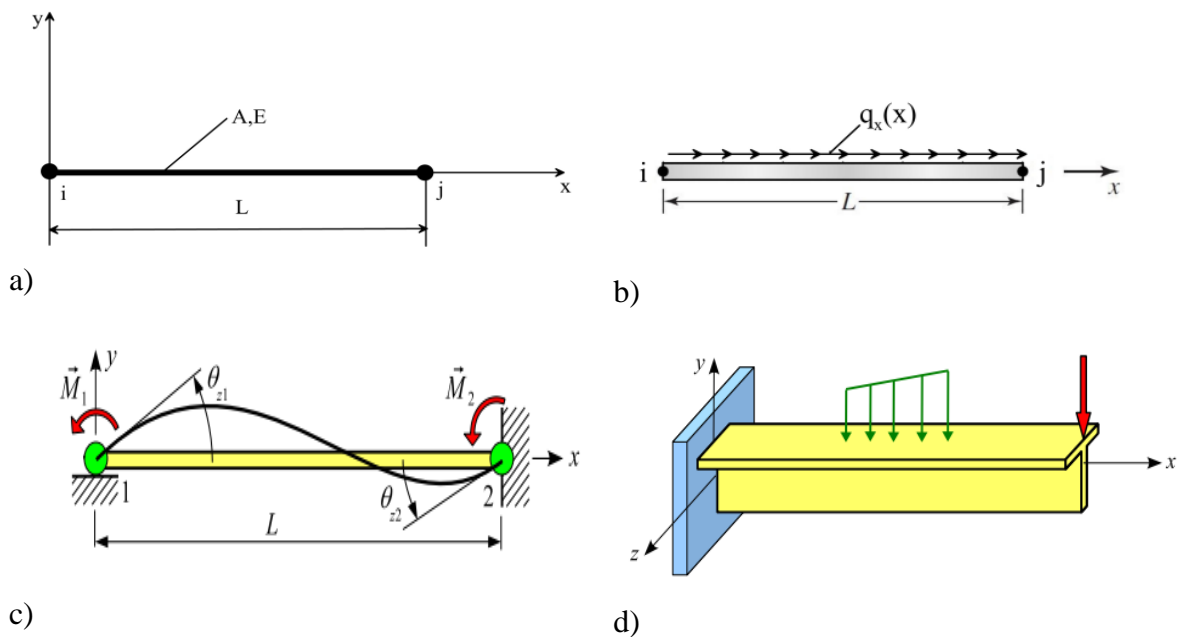
Advances in the structural analysis of airplanes led to the basic ideas of the finite element method. Hrennikoff offered by using “the frame work method” solution to elasticity problems that was in 1941. In 1956 Turner et al presented their findings which are derived stiffness matrices for truss, beam, and other elements. It was in 1960 that Clough formulated the term finite element and used it for the first time. Engineers were using it to approximate problems in stress analysis, fluid flow, heat transfer and other areas by the early 1960s. The foundation for further developments in finite element studies was set by a 1955 book by Argyris on energy theorems and matrix methods. Zienkiewicz and Cheung's first book on FEA was published in 1967. In the late 1960s and early 1970s; the application of finite element analysis was to non-linear problems and large deformations, in 1972. Oden's book on non-linear continua appeared. The mathematical foundations were laid in the 1970s. This category includes the development of new elements, convergence studies and other related areas. Today, the method is within reach of students and engineers working in small industries, thanks to the development of mainframe computers and the availability of powerful microcomputers [2].

### 1.2.2 Definition

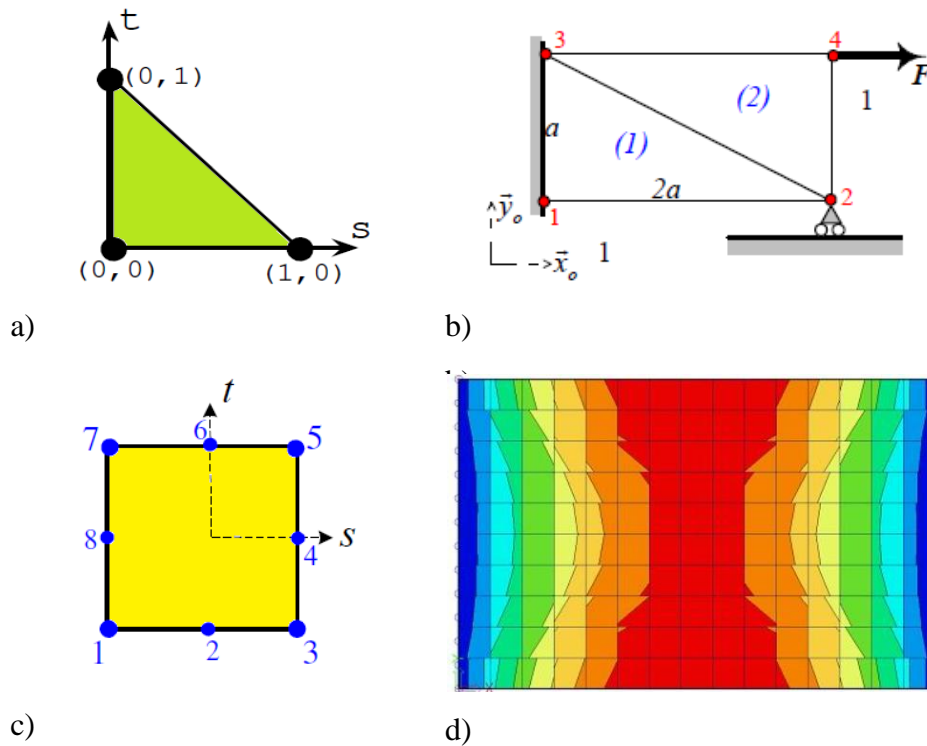
It is a numerical method for the solution of certain problems in physics and differential equations. It is a method of finding an approximate solution on a spatial domain, which calculates a field (of scalars, vectors, tensors) that satisfies certain equations and conditions [3]. It consists of dividing (discretizing) the domain of these equations into a number of sub domains, called finite elements, which are connected to each other by nodes (Figure 1.1 and Figure 1.2). The solution sought is replaced in each element by an approximation using simple polynomials. The domain can then be reconstructed by assembling all these elements. FEM applied to structural calculations is a multidisciplinary technique based on the following three disciplines:

- Structural mechanics: linear elasticity, resistance of materials (engineering science);
- Numerical analysis: methods of approximation, numerical integration, resolution of linear systems, etc;
- Computer science: development techniques (software engineering) and maintenance [4];

FEM is capable of solving 1D, 2D, and 3D systems, enabling the analysis of complex systems that are difficult or impossible to solve analytically.



**Figure 1.1:** Examples of 1D finite element: a) Bar element; b) Bar element subjected to a distributed load with density  $q(x)$ ; c) Beam element [5]; d) Beam element subjected to distributed load and concentrated force [5]



**Figure 1.2:**Examples of 2D finite elements: a) Triangular element with 3 nodes; b) Plate modeled by 2 triangular elements with 3 nodes; c) Quadrilateral element with 8 nodes; d) Meshing of a plate with quadrilateral elements

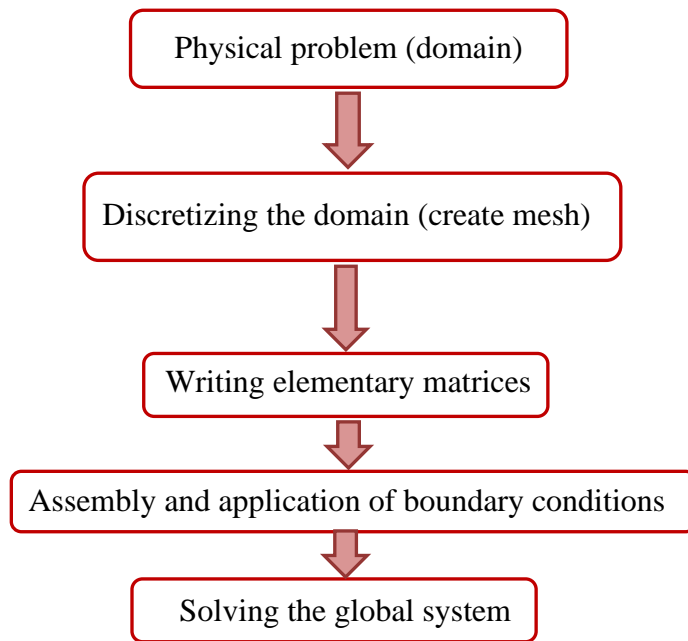
### 1.3 Steps of the Finite Element Method

This text outlines the steps involved in applying the finite element method and the tools required to implement it in a simplified manner. The process of solving a physical problem using finite elements generally follows these steps [6]:

- Discretizing the domain (create mesh): The first step in the discretization process is to subdivide the domain into elements and nodes. For systems that are already discrete, such as trusses, this step is unnecessary and the obtained answers are exact. However, for continuous systems such as plates, this step becomes critical and the answers obtained are only approximate. The accuracy of the solution in this case depends on the discretization used;
- Writing the element stiffness matrices: The element stiffness matrices must be written for each element in the domain;

- Assembling the global stiffness matrix: After wrote matrix for each element, we will assemble them by using the direct stiffness approach;
- Boundary conditions: Applying boundary conditions involves specifying supports, applied loads, and displacements;
- Solving the global system: The equations are solved by partitioning the global stiffness matrix. The resulting equations are then solved using Gaussian elimination.

The steps can be schematized in (Figure 1.3).



**Figure 1.3:** Flowchart of the various steps of the finite element method

#### **1.4 The benefits of Finite Element Method**

The Finite Element Method (FEM) offers several advantages that make it a widely used numerical technique for solving engineering and mathematical problems. Some of the main advantages of FEM are:

- FEM can handle complex geometries and boundary conditions, making it suitable for analyzing a wide range of engineering problems. Engineers can accurately simulate real-world scenarios by modeling irregular shapes and non-uniform material properties;
- FEM enables engineers to achieve a balance between computational efficiency and accuracy by allowing the use of different types of elements, such as triangles and quadrilaterals, selected based on the specific characteristics of the problem;

- By dividing the domain into smaller elements and using higher order interpolation functions, FEM provides accurate solutions to engineering problems. As the element size decreases, the solution approaches the exact solution of the continuous problem, resulting in more accurate outcomes;
- FEM is a versatile tool, applicable to a broad range of physical phenomena, including structural, heat transfer, fluid flow, electromagnetic and geotechnical problems. Because of its versatility, it is a valuable tool for the analysis of a wide variety of engineering problems in a wide range of industries;
- FEM enables engineers to conduct parametric analysis, studying the effects of varying input parameters, such as material properties, geometric dimensions, and boundary conditions, on the system's behavior. This allows for optimization, sensitivity analysis, and design exploration.

## 1.5 Truss systems

### 1.5.1 Definition

Truss systems are structures made up of rigid elements connected by nodes to form a grid of triangles or squares. These systems are commonly used in civil and mechanical engineering. Trusses are structures made up of several elastic bars that are subjected to axial forces only, meaning that the force in a straight section is reduced to a normal force. Trusses and Truss beams are crucial components in the construction industry. They are used to support floors, roofs, or provide bracing. These structures are essential for ensuring the stability and safety of bridges (Figure 1.4), buildings (Figure 1.5) [7].



**Figure 1.4:** Truss system in bridges



**Figure 1.5:** Trussing buildings

### 1.5.2 The principle of a truss system

The truss principle is simple. It consists of a top chord and a bottom chord connected by a triangulation of bars, with each bar only absorbing a normal load. While additional effects may exist, they are secondary in a well-designed truss. The chords absorb the overall moment in the form of compression or traction, while the diagonals take up the overall trench force in the form of compression or traction. In the simple case, when connections are treated as joints and loads are applied to the nodes, no bending moment, trench force or torsion is produced in any of the bars. If loads are applied in a way that produces bending moment, trench force or torsion, it will result in inefficient use of the material [7].

### 1.5.3 Advantages of truss system

The truss system is a commonly used structural design in bridges, buildings and race cars as shown in (Figure 1.6), due to its several advantages. This design offers a high strength-to-weight ratio, excellent stability, and efficient use of materials. Geometric simplicity is a key feature of these designs, which are typically composed of rectilinear bars. This facilitates their design and manufacturing processes. It provides a high level of structural strength despite its lightweight nature.



**Figure 1.6:** Example of application of truss system in the manufacturing of race cars

## 1.6 Beam element

### 1.6.1 Introduction

In structural engineering, beams are fundamental components used to support loads in buildings, bridges, and various infrastructure projects. Understanding the behavior of beams is crucial for designing safe and efficient structures.

### 1.6.2 Definition of beams

Beams are structural members primarily subjected to bending moments and shear forces (Figure 1.7). They typically have a slender and elongated shape compared to their cross-sectional dimensions. Beams are designed to resist loads applied perpendicular to their longitudinal axis while transferring these loads to their supports [10].

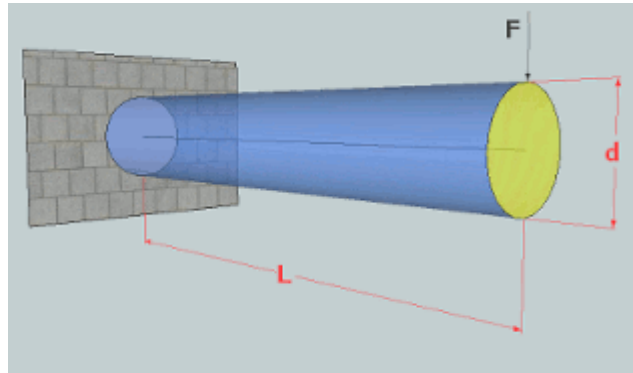


Figure 1.7: Schematic representation of a beam

### 1.6.3 Types of beams

#### 1.6.3.1 Simple beams

Simple beams are supported at both ends and subjected to loads along their length (Figure 1.8). They are the most basic type of beam configuration and are commonly used in building construction.

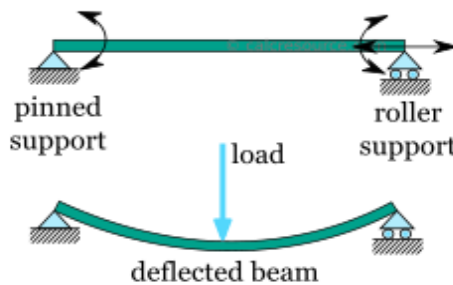
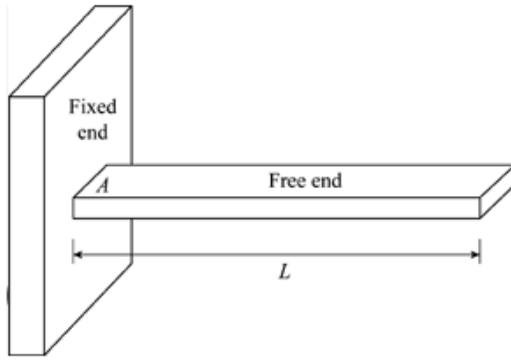


Figure 1.8: Schematic representation of a simple beam

#### 1.6.3.2 Cantilever beams

Continuous beams have multiple supports along their length, allowing them to carry heavier loads and span longer distances without additional intermediate supports (Figure 1.9 and Figure 1.10).





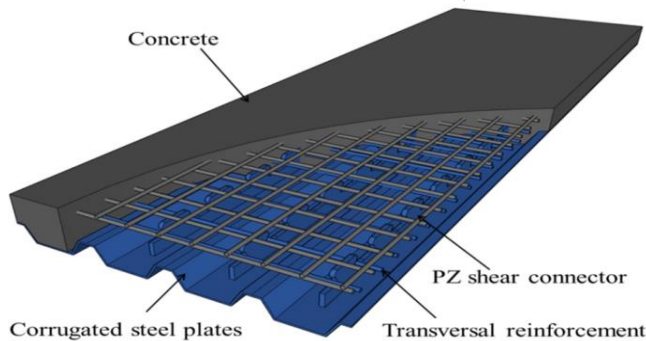
**Figure 1.9:** Schematic representation of cantilever beam



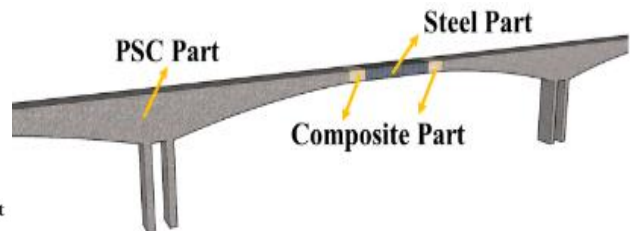
**Figure 1.10:** Example of application of cantilever beam

### 1.6.3.3 Composite beams

Composite beams are made from two or more different materials, such as steel and concrete, combined to enhance strength and stiffness (Figure 1.11 and Figure 1.12). They are frequently used in bridge construction to optimize structural performance.



**Figure 1.11:** Schematic representation of composite



**Figure 1.12:** Composite beams used in bridge construction

### 1.6.3.4 Truss beams

Truss beams consist of smaller interconnected members arranged in a triangular pattern (Figure 1.13). They are lightweight yet structurally efficient and are commonly used in roof and bridge designs.



**Figure 1.13:** Schematic representation of truss beam

## ***1.6.4 Flexural behavior of beams***

### ***1.6.4.1 Bending moment***

Beams undergo bending when subjected to transverse loads, resulting in the development of bending moments along their length. The magnitude of the bending moment varies along the beam's span and is influenced by the applied load and beam geometry.

### ***1.6.4.2 Shear force***

Shear forces act perpendicular to the longitudinal axis of the beam and arise due to transverse loading. They induce internal stresses that can cause shear deformation or failure in the beam.

### ***1.6.4.3 Bend***

Shear forces act perpendicular to the longitudinal axis of the beam and arise due to transverse loading. They induce internal stresses that can cause shear deformation or failure in the beam.

## ***1.6.5 Euler-Bernoulli Beam Theory***

The Euler-Bernoulli beam theory provides a fundamental framework for analyzing the flexural behavior of beams. It relates bending moment, shear force, and deflection to beam geometry and material properties, enabling engineers to predict beam performance accurately [10].

## ***1.7 Conclusion***

The Finite Element Method (FEM) is a powerful numerical technique used to determine the approximate equilibrium state of a continuous medium. It has the major advantage of being adaptable to solve a variety of problems with minimal modifications. FEM can be applied to any complex geometric domain where a problem is well-posed with all boundary conditions. However, proficiency in mathematical and computer tools is necessary to use this method

effectively [4]. In general, the finite element method provides engineers with a tough and versatile approach to solving complex engineering problems. It offers accurate, efficient solutions to a wide range of challenges.

## Chapter 2

# Finite element one dimensional 1D

### 2.1 Introduction

In the field of engineering and applied sciences, modeling of structures and physical phenomena is essential for understanding and predicting their behavior. Numerical methods, particularly finite element methods, play a crucial role in this modeling. The finite element method (FEM) uses one-dimensional (1D) finite elements to model physical phenomena that vary in only one spatial dimension. These elements are typically used to represent structures, components, or systems that exhibit behavior primarily along a single axis or direction. One-dimensional finite elements are geometrically simple and represent a line segment or curve along the one-dimensional axis. They typically have two nodes at each end, defining the nodes of the element. In the finite element method, a unidirectional domain is divided into linear elements and an equation is developed for each of these elements. This process will be extended to the entire domain to generate a continuum of equations for the entire domain of linear elements [3].

In this chapter, we will explore the fundamentals of one-dimensional (1D) finite elements. This approach simplifies modeling by reducing the problem to a single spatial dimension while retaining the ability to represent a wide variety of physical phenomena.

### 2.2 Fundamental concepts

#### 2.2.1 Domain discretization

The finite element approach involves discretizing the physical domain into smaller, simpler elements called finite elements. In one dimension, the domain is represented by a series of segments or intervals.

#### 2.2.2 Shape functions

To describe behavior within each finite element, shape functions are used. In one dimension, these functions are typically simple polynomials, such as linear or quadratic functions.

#### 2.2.3 Stiffness matrix

The stiffness matrix represents interactions between nodes in the mesh, taking into account material and geometric properties of the problem. In one dimension, this simplified matrix reflects stress-strain relationships within each element.

## 2.3 Solution procedure

### 2.3.1 Establishing equilibrium equations

Using the principle of virtual work or other energy minimization methods, equilibrium equations are established for the problem. In one dimension, these equations often boil down to ordinary differential equations.

### 2.3.2 Assembling the equation System

Local equations associated with each finite element are assembled to form a global system of equations. This system represents the entirety of the problem and can be solved numerically.

### 2.3.3 Applying boundary conditions

Boundary conditions, such as Dirichlet or Neumann conditions, are applied to represent system behavior at the boundaries of the domain.

## 2.4 Bar element

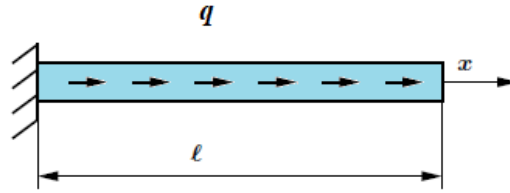
A bar element is a fundamental component used in structural analysis within the framework of the finite element method (FEM). It is used for the modeling of one-dimensional (1D) structures such as beams, columns, trusses and other members that are primarily subject to axial loads (traction or compression). The term 'bar' in this context refers to a slender structural element, often represented by a straight-line segment, along which deformation occurs primarily in the axial direction.

### 2.4.1 Governing equations

The finite element method (FEM) uses governing equations that are dependent on the type of physical problem being solved.

### 2.4.2 Equilibrium equations

These equations express the balance of forces and moments acting on a structure. They are derived from Newton's second law of motion and are represented as equilibrium equations for each node or element in the finite element model. To formulate this element, we consider a bar with constant cross-sectional area  $\mathbf{A}$  and length  $\mathbf{L}$ , subjected to a tensile load  $\mathbf{q}$  (N/m)(Figure 2.1). The bar exhibits linear elastic behavior (Young's modulus  $\mathbf{E}$ ). We seek the displacement field  $\mathbf{u}$ , strain  $\boldsymbol{\epsilon}$ , and stress  $\boldsymbol{\sigma}$  at all points along the bar.



**Figure 2.1:** Schematic representation of a bar, subjected to a tensile load  $q$

The equation is obtained by writing the equilibrium of forces for an elemental segment of the bar with length  $dx$ , located at a distance  $x$  from the edge 0 (Figure 2.2).

$$N + \frac{dN}{dx} dx - N + q dx = 0 \quad (2.1)$$

So:

$$\frac{dN}{dx} + q = 0 \quad (2.2)$$

The internal axial force  $N(x)$  can, by application of Hooke's law, be expressed as a function of the displacement  $u$ :

$$\sigma_x = E \varepsilon_x \quad (2.3)$$

$$\varepsilon_x = \frac{du}{dx} \quad (2.4)$$

$$\sigma_x = E \frac{du}{dx} \quad (2.5)$$

Knowing that:

$$N(x) = A \sigma_x \quad (2.6)$$

So:

$$N(x) = AE \frac{du}{dx} \quad (2.7)$$

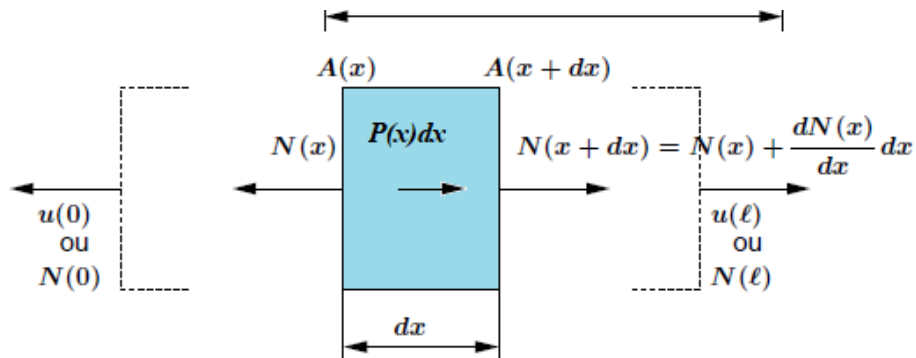
Substituting (27) into (2.2), the equilibrium equation of the bar (one dimension) and the boundary conditions are finally written as:

For  $0 < x < l$

$$AE \frac{d^2u}{dx^2} + q = 0$$

$$N(L) = \left( \frac{du}{dx} \right)_{x=L} = 0 \quad \text{for } x = L \quad (2.8)$$

$$u = 0 \quad \text{for } x = 0$$



**Figure 2.2:** Equilibrium of forces for an elemental segment of the bar with length  $dx$

### 2.4.3 Approximate solution using the finite element method

#### 2.4.3.1 General integral formulation

Let  $\mathbf{u}(\mathbf{x})$  is an interpolation function of the displacement function, denoted as  $\bar{u}(\mathbf{x})$ . We form the residual  $\mathbf{R}$  from the equilibrium equation (2.7), multiply it by the weighting function  $\phi$  that satisfies the displacement boundary conditions, and integrate over the study domain (interval  $[0, L]$ ). The cross-section of the bar is constant:

$$\int_0^L \phi \left( AE \frac{d^2 \bar{u}}{dx^2} + q \right) dx = 0 \quad (2.9)$$

#### 2.4.3.2 Weak integral formulation

To simplify this integral (2.9), we perform integration by parts on the first term of this integral, as follows:

$$\int_0^L \phi \frac{d^2 \bar{u}}{dx^2} dx = \left[ \frac{d\bar{u}}{dx} \right]_0^L - \int_0^L \left( \frac{d\phi}{dx} \frac{d\bar{u}}{dx} \right) dx = 0 \quad (2.10)$$

This expression is based on the integration by parts formula, which states as follows:

$$(uv)' = u'v + uv' \quad (2.11)$$

By introducing the boundary conditions of this problem (Eq.2.8):

$$\left( \frac{d\bar{u}}{dx} \right)_{x=L} = 0 \quad (2.12)$$

And:

$$\bar{u}(x=0) = 0 \quad (2.13)$$

We can write:

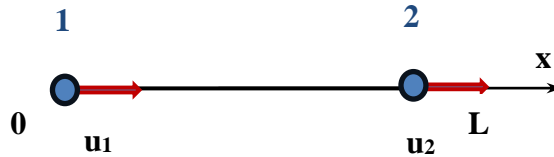
$$\int_0^L \phi \frac{d^2 \bar{u}}{dx^2} dx = - \int_0^L \left( \frac{d\phi}{dx} \frac{d\bar{u}}{dx} \right) dx = 0 \quad (2.14)$$

By substituting into the Equation 2.9, we obtain:

$$- \int_0^L AE \left( \frac{d\phi}{dx} \frac{d\bar{u}}{dx} \right) dx + \int_0^L q \phi dx = 0 \quad \forall \phi \quad (2.15)$$

### 2.4.3.3 Discretization

For the discretization, we use the Galerkin method (the test function is taken to be equal to the variation of the displacement ( $\phi = \delta u$ )). The study domain is discretized using a two-node bar element of length  $L$  (Figure 2.3).



**Figure 2.3:** Schematic representation of a two-node linear bar finite element

This means the displacement within the element is interpolated based on the displacements at the two ends of the bar. Since the interpolation is linear, the displacement for a bar element (nodes **1** and **2**) can be expressed as:

$$\bar{u}(x) = \langle N(x) \rangle \{ u_n \} = \langle N_1(x) \quad N_2(x) \rangle \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \langle u_n \rangle \{ N(x) \} = \langle u_1 \quad u_2 \rangle \begin{Bmatrix} N_1(x) \\ N_2(x) \end{Bmatrix} \quad (2.16)$$

$u_n$  represents the nodal values of the approximation function  $\bar{u}(x)$ . The functions  $N_1(x)$  and  $N_2(x)$  are the shape functions constructed to satisfy the following conditions:

$$\begin{aligned} \bar{u}(0) &= u_1 \\ \bar{u}(L) &= u_2 \end{aligned} \quad (2.17)$$

The linear interpolation functions for a two-node bar element (nodes **1** and **2**) of length  $L = x_2 - x_1$  are defined as follows:



$$N_1(x) = \frac{x - x_2}{x_2 - x_1} = 1 - \frac{x}{L} \quad (2.18)$$

And:

$$N_2(x) = \frac{x - x_1}{x_2 - x_1} = \frac{x}{L} \quad (2.19)$$

By using these functions and the relation (2.16), we can determine the displacement at any point of the element. If we want to replace  $\bar{u}(x)$  in Equation (2.15), we need to evaluate  $\frac{d\bar{u}}{dx}$ :

$$\frac{d\bar{u}}{dx} = \frac{d\langle N(x) \rangle \{u_n\}}{dx} = \langle \frac{dN(x)}{dx} \rangle \{u_n\} = \langle u_n \rangle \left\{ \frac{dN(x)}{dx} \right\} \quad (2.20)$$

From the Equations (2.18) and (2.19), we have:

$$\frac{dN_1(x)}{dx} = -\frac{1}{L} \quad (2.21)$$

And:

$$\frac{dN_2(x)}{dx} = \frac{1}{L} \quad (2.22)$$

By using Equation (2.20), the relation (2.15), becomes:

$$-\int_0^L AE \langle \phi_n \rangle \left\{ \frac{dN(x)}{dx} \right\} \left\langle \frac{dN(x)}{dx} \right\rangle \{u_n\} dx + \int_0^L q \langle \phi_n \rangle \{N(x)\} dx = 0 \quad (2.23)$$

Since  $\phi_n$  and  $u_n$  are not functions that depend on the integration variable  $x$ , we can write that:

$$\langle \phi_n \rangle \left( -\int_0^L AE \left\{ \frac{dN(x)}{dx} \right\} \left\langle \frac{dN(x)}{dx} \right\rangle \{u_n\} dx + q \int_0^L \{N(x)\} dx \right) = 0 \quad (2.24)3$$

This relation is verified regardless of the value of the function  $\phi_n$  so we can write that:

$$\left( -\int_0^L AE \left\{ \frac{dN(x)}{dx} \right\} \left\langle \frac{dN(x)}{dx} \right\rangle dx \{u_n\} + q \int_0^L \{N(x)\} dx \right) = 0 \quad (2.25)3$$

We initiate the process by computing the components of the integral (2.25)

$$\left\{ \frac{dN(x)}{dx} \right\} \left\langle \frac{dN(x)}{dx} \right\rangle = \begin{Bmatrix} \frac{dN_1(x)}{dx} \\ \frac{dN_2(x)}{dx} \end{Bmatrix} \left\langle \frac{dN_1(x)}{dx} \quad \frac{dN_2(x)}{dx} \right\rangle = \frac{1}{L^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (2.26)$$

The different integrals can be evaluated using interpolation functions, as follows:

$$\int_0^L \text{AE} \left\{ \frac{dN(x)}{dx} \right\} \left\langle \frac{dN(x)}{dx} \right\rangle dx = \int_0^L \frac{\text{AE}}{L^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx = \frac{\text{AE}}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (2.27)3$$

Additionally, we have:

$$\int_0^L \text{AE} \{N(x)\} dx = \int_0^L \begin{Bmatrix} 1 - \frac{x}{L} \\ \frac{x}{L} \end{Bmatrix} dx = \begin{bmatrix} [x]_0^L - \left[ \frac{x^2}{2L} \right]_0^L \\ \left[ \frac{x^2}{2L} \right]_0^L \end{bmatrix} = \frac{L}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \quad (2.28)3$$

By substituting in Equation (2.25), we get:

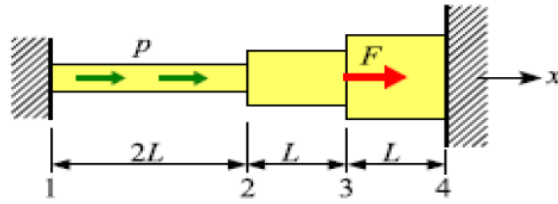
$$\frac{\text{AE}}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \frac{ql}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \quad (2.29)$$

The discretized Equation takes the form:

$$[K] \{u_n\} = \{f_n\} \quad (2.30)$$

#### 2.4.4 Example of the calculation for a bar element in tension

To illustrate the use of one-dimensional finite elements, we will examine a practical application Example. We need to analyze the behavior of a beam subjected to tensile force using the finite element method. The beam with the x-axis shown in the (Figure 2.4) is made of a material with Young's modulus (E). The cross-sectional area is equal to (**A**) between nodes 1 and 2, (**2A**) between nodes 2 and 3, and (**3A**) between nodes 3 and 4. The beam is fixed at nodes 1 and 4, and it carries a uniformly distributed load of linear intensity (**p**) between nodes 1 and 2, and a force (**F=2pL**) at node 3. We aim to determine the nodal displacements and the support reactions.



**Figure 2.4:** Example of the calculation of a beam subjected to tensile force

Given:

$$E=2 \times 10^5 \text{ Mpa}$$

$$A=200 \text{ mm}^2$$

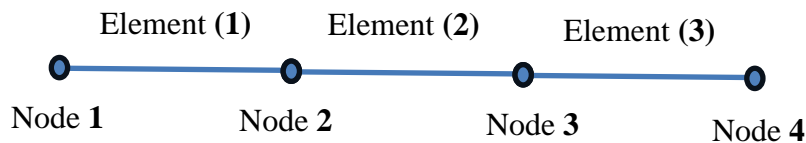
$$p= 100 \text{ N/m}$$

$$L=1000 \text{ mm}$$

The steps to solve this problem are as follows:

**1. Model of the beam structure:** The geometry and material properties of the beam are presented in [Figure 2.4](#).

**2. Discretize of the beam:** Divide the beam into three finite elements between the nodes. Identify the nodes and elements, with nodes 1, 2, 3, and 1,2,3,4 as shown in [Figure 2.5](#):



**Figure 2.5:** Discretization of the beam into 3 linear elements and 4 nodes

**3. Set up of the element stiffness matrices:** Calculate the stiffness matrix for each element using the formula for a beam element in the finite element method (FEM).

- **Element 1:** between nodes 1 and 2, with cross-section A, Young's modulus E, and length 2L.

$$[K]^{(1)} = \left( \frac{EA}{L} \right)^{(1)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{EA}{2L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

- **Element 2:** between nodes 2 and 3, with cross-section 2A, Young's modulus E, and length L.

$$[K]^{(2)} = \left( \frac{EA}{L} \right)^{(2)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{2EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$$

- **Element 3:** between nodes 3 and 4, with cross-section 3A, Young's modulus E, and length L.

$$[K]^{(3)} = \left(\frac{EA}{L}\right)^{(3)} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{3EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 3 & -3 \\ -3 & 3 \end{bmatrix}$$

**4. Assemble the global stiffness matrix:** Combine the element stiffness matrices to form the global stiffness matrix (GSM).

Dim (GSM) = number of degrees of freedom per node multiplied by the number of nodes

Dim (GSM) =  $1 \times 4 = 4$

$$[K] = \frac{EA}{L} \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & \frac{5}{2} & -2 & 0 \\ 0 & -2 & 5 & -3 \\ 0 & 0 & -3 & 3 \end{bmatrix}$$

**5. Formulate the load vector:** Create the global load vector  $\{F\}$  based on the applied concentrated force ( $F=2pL$ ) at node 3 and the uniformly distributed load  $p$  between nodes 1 and 2.

- **Element 1:** between nodes 1 and 2, uniformly distributed load  $P$

$$\{f\}^{(1)} = \left(\frac{pL}{2}\right)^{(1)} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} = pL \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$$

- **Element 2:** between nodes 2 and 3, concentrated force at node 3,  $F_2=0$  and  $F_3=pL$

$$\{f\}^{(2)} = \begin{Bmatrix} 0 \\ pL \end{Bmatrix}$$

- **Element 3:** between nodes 3 and 4, concentrated force at node 3,  $F_3=pL$  and  $F_4=0$

$$\{f\}^{(3)} = \begin{Bmatrix} pL \\ 0 \end{Bmatrix}$$

**6. Global force vector:** The vector that represents the cumulative forces applied at the nodes of the entire finite element mesh. It is assembled by summing the contributions of all local force vectors, thereby encompassing all the forces acting across the entire structure.

$$\{F\}^T = \{pL \quad pL \quad 2pL \quad 0\}$$

**7. Applying boundary conditions:** The boundary conditions specific to this case study are presented in [Table 2.1](#).

**Table 2.1:** The boundary conditions specific to the case study (bar in tension)

Nodes	Boundary conditions	Displacements	Forces
Node 1	fixed support	$u_1 = 0$	$F_1 = pL$ and $R_1 = ?$
Node 2	free node	$u_2 = ?$	$F_2 = pL$
Node 3	concentrated force	$u_3 = ?$	$F_3 = 2pL$
Node 4	fixed support	$u_4 = 0$	$R_4 = ?$

**8. Formation of the global system:** Solving the system of equations to calculate displacements and the reactions at the supports (nodes 1 and 4) using the relationship between the stiffness matrix, displacements, and external forces.  $[K] \cdot \{U\} = \{F\}$ .

$$\frac{EA}{L} \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & \frac{5}{2} & -2 & 0 \\ 0 & -2 & 5 & -3 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix} = \begin{Bmatrix} pL \\ pL \\ 2pL \\ 0 \end{Bmatrix} + \begin{Bmatrix} R_1 \\ 0 \\ 0 \\ R_4 \end{Bmatrix}$$

**Solution:** To solve this system, we eliminate the rows and columns with zero displacements, namely 1 and 4, resulting in the reduced system:

$$\frac{EA}{L} \begin{bmatrix} \frac{5}{2} & -2 \\ -2 & 5 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \end{Bmatrix} = \begin{Bmatrix} pL \\ 2pL \end{Bmatrix}$$

We use Cramer's method to solve this system of equations with two unknowns:

$$\Delta = \begin{vmatrix} \frac{5}{2} \frac{EA}{L} & -2 \frac{EA}{L} \\ -2 \frac{EA}{L} & 5 \frac{EA}{L} \end{vmatrix} = \frac{17E^2A^2}{2L^2}$$

$$\Delta_{u_2} = \begin{vmatrix} pL & -2 \frac{EA}{L} \\ 2pL & 5 \frac{EA}{L} \end{vmatrix} = 9EA p$$

$$\Delta u_3 = \begin{vmatrix} \frac{5}{2} \frac{EA}{L} & pL \\ -2 \frac{EA}{L} & 2pL \end{vmatrix} = 7EA p$$

Finally, we find:

$$u_2 = \frac{\Delta u_2}{\Delta} = \frac{18pL^2}{17EA}$$

$$u_3 = \frac{\Delta u_3}{\Delta} = \frac{14pL^2}{17EA}$$

To determine the reactions at the supports, we reconsider equations 1 and 4 and replace the displacements  $u_2$  and  $u_3$  with their previously determined values:

$$\frac{EA}{L} \left( \frac{-1}{2} u_2 \right) = pL + R_1 = \frac{-1}{2} \frac{EA}{L} \left( \frac{18pL^2}{17EA} \right) = pL + R_1 = \frac{-9pL}{17} \Rightarrow R_1 = \frac{-26}{17} pL$$

$$\frac{EA}{L} (-3u_3) = R_4 = -3 \frac{EA}{L} \left( \frac{14pL^2}{17EA} \right) = R_4 = \frac{-42pL}{17} \Rightarrow R_4 = \frac{-42}{17} pL$$

Numerical Application: After calculation, the results for the unknown displacements and reactions are shown in the following Table 2.2:

**Table 2.2:** Results specific to the case study (bar in tension)

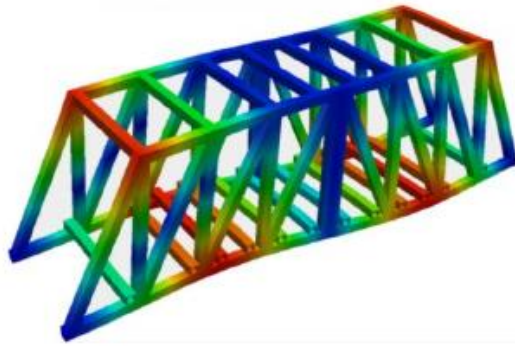
Nodes	Displacements	Forces
Node 1	$u_1 = 0$	$F_1 = 100 \text{ N}$ and $R_1 = -152.94 \text{ N}$
Node 2	$u_2 = 0.0026 \text{ mm}$	$F_2 = 100 \text{ N}$
Node 3	$u_3 = 0.0021 \text{ mm}$	$F_3 = 200 \text{ N}$
Node 4	$u_4 = 0$	$R_4 = -247.06 \text{ N}$

## 2.5 Truss element

### 2.5.1 Definition of truss systems

Truss systems are structures made up of rigid elements connected by nodes, forming a network of triangles or quadrilaterals. These structures are widely used in various fields and sectors of engineering, particularly in civil engineering and mechanical engineering. Trusses are structures composed of several elastic bars subjected only to axial forces. The force in a straight section is reduced to a normal force. A classic example of a truss system with triangular elements is the Pratt truss bridge (Figure 2.6). This type of bridge uses a truss configuration with triangles

to offer great strength and stability. The diagonal elements are inclined to support vertical loads and distribute forces efficiently throughout the structure [7].



**Figure 2.6:** Schematic representation of a truss system with triangular elements (Pratt truss bridge)

### 2.5.2 Characteristics of truss systems

Truss systems have several advantages, including:

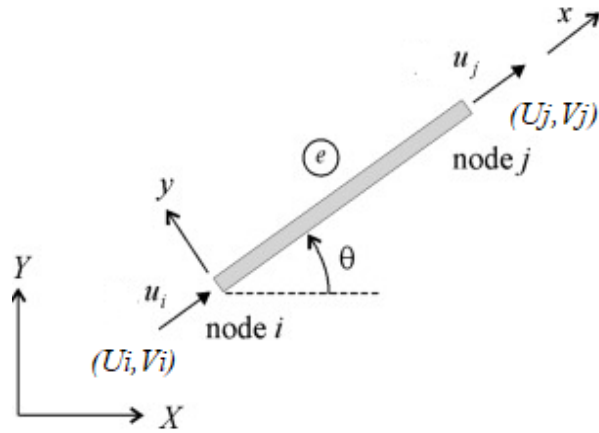
- **Geometric simplicity:** Generally composed of straight bars, which simplifies their design and fabrication.
- **Lightweight:** Trusses provide high structural strength while being relatively light and resistant

### 2.5.3 Formulation of the truss finite element

The elemental stiffness matrix of a truss element is a mathematical representation that describes the relationship between the nodal displacements and the resulting forces within the element. It is a key component in the finite element analysis of truss structures, allowing for the determination of internal forces and deformations under applied loads. The stiffness matrix encapsulates the geometric and material properties of the truss element, ensuring that the overall structural behavior is accurately modeled. The formulation of the elemental stiffness matrix for a truss element is a key step in the finite element analysis of truss structures. The basic assumptions of the formulation of a truss element are:

- The bars are considered as rigid 1D elements;
- Forces are applied at the nodes;
- Deformations are small (small deformation hypothesis);

Consider a bar element (truss structure element with two nodes (i) and (j), inclined at an angle  $\theta$  relative to the global coordinate system as shown in (Figure 2.7) [8].



**Figure 2.7:** Schematic representation of truss element with two nodes (i) and (j), inclined at an angle  $\theta$

$(X, Y)$  represents the global system.  $(x, y)$  represents the local system. In the local coordinate system, this element has 2 degrees of freedom  $(\mathbf{u}_i, \mathbf{v}_j)$ , i.e., 1 degree of freedom per node. In the global coordinate system, this element has 4 degrees of freedom  $(U_i, V_i, U_j, V_j)$ , i.e., 2 degrees of freedom per node. The projection of displacement vectors from the local coordinate system to the global coordinate system is as follows:

$$u_i = U_i \cos \theta + V_i \sin \theta \quad (2.31)$$

And:

$$u_j = U_j \cos \theta + V_j \sin \theta \quad (2.32)$$

With:

$$\sin \theta = \frac{Y_j - Y_i}{L} \quad (2.33)$$

And:

$$\cos \theta = \frac{X_j - X_i}{L} \quad (2.34)$$

The expression of equations (2.31) and (2.32) in matrix form leads to:

$$\begin{Bmatrix} \mathbf{u}_i \\ \mathbf{u}_j \end{Bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ 0 & 0 & \cos \theta & \sin \theta \end{bmatrix} \begin{Bmatrix} U_i \\ V_i \\ U_j \\ V_j \end{Bmatrix} \quad (2.35)$$



So, we have:

$$\{u\} = [T_e] \cdot \{U\} \quad (2.36)$$

In this relation,  $\{u\}$  represents the displacement vector in the local coordinate system,  $\{U\}$  represents the displacement vector in the global coordinate system, and  $[T_e]$  is the transformation matrix between the two systems. The same notation can be used for expressing the forces acting on the two nodes:

$$\begin{Bmatrix} f_i \\ f_j \end{Bmatrix} = \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ 0 & 0 & \cos \alpha & \sin \alpha \end{bmatrix} \begin{Bmatrix} F_{ix} \\ F_{iy} \\ F_{jx} \\ F_{jy} \end{Bmatrix} \quad (2.37)$$

So:

$$\{f\} = [T_e] \cdot \{F\} \quad (2.38)$$

In equation (2.38),  $\{f\}$  represents the force vector in the local coordinate system, while  $\{F\}$  represents the force vector in the global coordinate system.  $[T_e]$  represents the transformation matrix between the two systems. The stiffness law for this system is written (with respect to the global coordinate system):

$$[K] \cdot \{U\} = \{F\} \quad (2.39)$$

And with respect to the local coordinate system:

$$[k] \cdot \{u\} = \{f\} \quad (2.40)$$

If we substitute the relations (2.36) and (2.38) into equation (2.40), we obtain:

$$[k] \cdot [T_e] \cdot \{U\} = [T_e] \cdot \{F\} \quad (2.41)$$

We multiply both sides of this equation by  $[T_e]^{-1}$ :

$$[T_e]^{-1} \cdot [k] \cdot [T_e] \cdot \{U\} = [T_e]^{-1} \cdot [T_e] \cdot \{F\} \quad (2.42)$$

After simplification, this relation becomes:

$$[T_e]^{-1} \cdot [k] \cdot [T_e] \cdot \{U\} = \{F\} \quad (2.43)$$

The transformation matrix is orthogonal, so:

$$[T_e]^{-1} = [T_e]^T \quad (2.44)$$

Substituting (2.44) into equation (2.43), we get:

$$[T_e]^T \cdot [k] \cdot [T_e] \cdot \{U\} = \{F\} \quad (2.45)$$

The general form of the stiffness  $[K] \cdot \{U\} = \{F\}$  law becomes apparent. This allows us to deduce:

$$K_e = [T_e]^T \cdot [k] \cdot [T_e] \quad (2.46)$$

$[k]$  represent the stiffness matrix of the element with respect to the local coordinate system where the relationship is given as follows:

$$[k] = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix} \quad (2.47)$$

If we substitute equations (2.47) into (2.46), we obtain:

$$[k]^{(e)} = \begin{bmatrix} \cos \alpha & 0 \\ \sin \alpha & 0 \\ 0 & \cos \alpha \\ 0 & \sin \alpha \end{bmatrix} \cdot \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix} \cdot \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ 0 & 0 & \cos \alpha & \sin \alpha \end{bmatrix} \quad (2.49)$$

To simplify the expression, let's establish,  $C = \cos \alpha$  and  $S = \sin \alpha$ . After transformation, equation (2.49) becomes:

$$[K] = \frac{EA}{L} \begin{bmatrix} \cos \alpha & -\cos \alpha \\ \sin \alpha & -\sin \alpha \\ -\cos \alpha & \cos \alpha \\ -\sin \alpha & \sin \alpha \end{bmatrix} \begin{bmatrix} \cos \alpha & \sin \alpha & 0 & 0 \\ 0 & 0 & \cos \alpha & \sin \alpha \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} \quad (2.50)$$

Finally, the expression for the elemental stiffness matrix of a truss system bar element is:

$$[k]^{(e)} = \frac{EA}{L} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} \quad (2.51)$$

We define:

$$[A] = \begin{bmatrix} C^2 & CS \\ CS & S^2 \end{bmatrix} \quad (2.52)$$

Equation (2.50) becomes:

$$[K_e] = \frac{EA}{L} \begin{bmatrix} [A] & -[A] \\ -[A] & [A] \end{bmatrix} \quad (2.53)$$

#### 2.5.4 Stress State of a Truss Finite Element

The stress state of a truss finite element refers to the internal forces that develop within the element in response to applied loads. These stresses are generally axial, as truss elements are

designed to primarily support tensile or compressive forces along their axis. The basic assumptions adopted for analyzing the stresses are as follows:

- **Linear Elasticity:** The materials behave in a linearly elastic manner up to the elastic limit;
- **Axial Loads:** The bars support only axial forces (tension or compression);
- **Homogeneity and isotropy:** The materials of the bars are homogeneous and isotropic;

According to Equation (2.35), in the local coordinate system, the coordinates of the displacement vector are defined as follows:

$$\{u\} = \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & 0 & 0 \\ 0 & 0 & \cos \theta & \sin \theta \end{bmatrix} \begin{Bmatrix} U_i \\ V_i \\ U_j \\ V_j \end{Bmatrix}$$

Given that the stress state (1D) in the local coordinate system is defined as follows:

$$N = \sigma S = ES \varepsilon(x) = ES \left( \frac{\partial U}{\partial x} \right) = ES \left\langle \frac{dN(x)}{dx} \right\rangle \{U_n\} = ES \left\langle \frac{-1}{L} \quad \frac{1}{L} \right\rangle \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = \frac{ES}{L} (u_i - u_j) \quad (2.54)$$

The stress state (2D) in the global coordinate system becomes:

$$N = ES \left\langle \frac{-1}{L} \quad \frac{1}{L} \right\rangle \begin{Bmatrix} u_i \\ u_j \end{Bmatrix} = ES \left\langle \frac{-1}{L} \quad \frac{1}{L} \right\rangle \begin{Bmatrix} U_i \cos \theta + V_i \sin \theta \\ U_j \cos \theta + V_j \sin \theta \end{Bmatrix} \quad (2.55)$$

Expanding this relation, we obtain:

$$N = \frac{ES}{L} \langle -1 \quad 1 \rangle \begin{Bmatrix} U_i \cos \theta + V_i \sin \theta \\ U_j \cos \theta + V_j \sin \theta \end{Bmatrix} \quad (2.56)$$

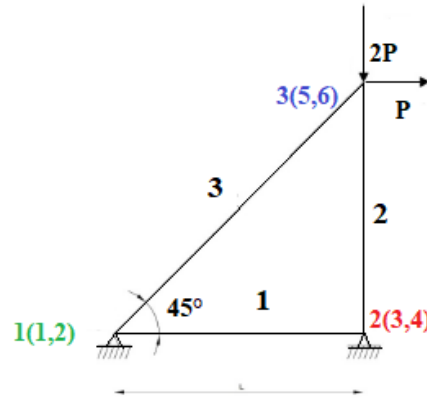
Finally:

$$N = \frac{ES}{L} (-U_i \cos \theta - V_i \sin \theta + U_j \cos \theta + V_j \sin \theta) = \frac{ES}{L} \langle \cos \theta \quad \sin \theta \rangle \begin{Bmatrix} U_j - U_i \\ V_j - V_i \end{Bmatrix} \quad (2.57)$$

### 2.5.5 Illustration demonstrating the computation for a truss finite element

To exemplify the application of one-dimensional finite elements, we'll explore a practical case study. Our objective is to analyze the response of a truss structure under tensile loading utilizing the finite element method. The truss system consists of 3 bars with the same stiffness  $E$  and cross-sectional area  $A$  (Figure 2.8). We aim to analyze the system's response by determining the unknown displacements and support reactions.

**Given:**  $P = 10 \text{ KN}$ ,  $L = 1 \text{ m}$ ,  $H = 1 \text{ m}$ ,  $A = 200 \text{ mm}^2$ ,  $E = 2 \times 10^5 \text{ MPa}$



**Figure 2.8:** Example of the calculation of a truss structure under tensile loading

The following steps outline the solution process:

**1. Properties of the structure:** the structure's geometry characteristics are summarized in the next [Table 2.3](#):

**Table 2.3:** The structure's geometry characteristics (truss structure under tensile loading)

Elements	Length (m)	Angle (°)	C	S	C <sup>2</sup>	S <sup>2</sup>	CS
Element (1)	1	0	1	0	1	0	0
Element (2)	1	90	0	1	0	1	0
Element (3)	$\sqrt{2}$	45	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

**2. Discretize of the structure:** Partition the structure into three finite elements spanning between the nodes. Define the nodes and elements, denoting them as nodes 1(1, 2), 2(3, 4), 3(5, 6), and elements 1, 2, 3 as illustrated in [Figure 2.8](#).

**3. Element stiffness matrices:** Compute the stiffness matrix for each element utilizing the formula for a beam element within the finite element method (FEM).

- **Element 1:** Spanning between nodes 1(1,2) and 2(3,4), with a cross-section A, Young's modulus E, and length L:

$$[\mathbf{K}]^{(1)} = \left( \frac{EA}{L} \right)^{(1)} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- **Element 2:** between nodes 2(3,4) and 3(5,6), with cross-section A, Young's modulus E, and length L.

$$[\mathbf{K}]^{(2)} = \left(\frac{EA}{L}\right)^{(2)} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

- **Element 3:** between nodes 1(1, 2) and 3(5, 6), with cross-section A, Young's modulus E, and length  $\sqrt{2}L$ .

$$[\mathbf{K}]^{(3)} = \left(\frac{EA}{L}\right)^{(3)} \begin{bmatrix} C^2 & CS & -C^2 & -CS \\ CS & S^2 & -CS & -S^2 \\ -C^2 & -CS & C^2 & CS \\ -CS & -S^2 & CS & S^2 \end{bmatrix} = \left(\frac{EA}{\sqrt{2}L}\right) \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

$$= \frac{EA}{L} \begin{bmatrix} \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} \\ -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \\ -\frac{\sqrt{2}}{4} & -\frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} & \frac{\sqrt{2}}{4} \end{bmatrix}$$

**4. Assemble the global stiffness matrix:** assemble the global stiffness matrix (GSM) by merging the stiffness matrices of the individual elements.

$$\text{Dim (GSM)} = n(\text{ddl})/n^*(\text{nn}) = 2 \times 3 = 6$$

After assembly and calculation, the final expression of the global stiffness matrix is as follows:

$$[\mathbf{K}] = 4 \times 10^4 \begin{bmatrix} 1.35 & 0.3536 & -1 & 0 & -0.3536 & -0.3536 \\ 0.3536 & 0.3536 & 0 & 0 & -0.3536 & -0.3536 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ -0.3536 & -0.3536 & 0 & 0 & 0.3536 & 0.3536 \\ -0.3536 & -0.3536 & 0 & -1 & 0.3536 & 1.3536 \end{bmatrix}$$

**5. Formulate the Load Vector:** Construct the global load vector  $\{F\}$  by considering the concentrated force (P, 2P) applied at node 3. The other nodes, 1 and 2, are free:

- **Element 1:** between nodes 1 and 2, no loading.

$$\{f\}^{(1)} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

- **Element 2:** between nodes 2 and 3, concentrated force at node 3,  $F_{3x} = \frac{P}{2}$  and  $F_{3y} = -P$

$$\{f\}^{(2)} = \begin{Bmatrix} 0 \\ 0 \\ \frac{P}{2} \\ 2 \\ -P \end{Bmatrix} \begin{matrix} 2 \\ 3 \\ 5 \\ 6 \end{matrix}$$

- **Element 3:** between nodes 1 and 3, concentrated force at node 3,  $F_{3x} = \frac{P}{2}$  and  $F_{3y} = -P$

$$\{f\}^{(3)} = \begin{Bmatrix} 0 \\ 0 \\ \frac{P}{2} \\ 2 \\ -P \end{Bmatrix} \begin{matrix} 1 \\ 2 \\ 5 \\ 6 \end{matrix}$$

**6. Global force vector:** The vector representing the total forces applied at the nodes throughout the entire finite element mesh is created by combining all local force vectors. This process includes all the forces acting on the entire structure.

Dim (GVF) = number of degrees of freedom per node multiplied by the number of nodes

Dim (GVF) =  $2 \times 3 = 6$

$$\{F\} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ P \\ -2P \end{Bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}$$

**7. Applying boundary conditions:** The boundary conditions for this particular case study are detailed in [Table 2.4](#):

**Table 2.4:** The boundary conditions specific to the case study (truss structure under tensile loading)

Nodes	Boundary conditions	Displacements	Forces
Node 1	fixed support	$u_1 = 0$ and $v_1 = 0$	$R_{1x} = ?$ and $R_{1y} = ?$
Node 2	fixed support	$u_2 = 0$ and $v_2 = 0$	$R_{2x} = ?$ and $R_{2y} = ?$
Node 3	concentrated force	$u_3 = ?$ and $v_3 = ?$	$F_{3x} = P$ and

**8. Formation of the global system:** Solving the system of equations to determine the displacements at node 3 and the reactions at the supports (nodes 1 and 2) using the relationship between the stiffness matrix, displacements, and external forces:  $[K] \cdot \{U\} = \{F\}$

$$\frac{EA}{L} \begin{bmatrix} 1.35 & 0.3536 & -1 & 0 & -0.3536 & -0.3536 \\ 0.3536 & 0.3536 & 0 & 0 & -0.3536 & -0.3536 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ -0.3536 & -0.3536 & 0 & 0 & 0.3536 & 0.3536 \\ -0.3536 & -0.3536 & 0 & -1 & 0.3536 & 1.3536 \end{bmatrix} \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ u_3 \\ v_3 \end{Bmatrix} = \begin{Bmatrix} R_{x1} \\ R_{y1} \\ R_{x2} \\ R_{y2} \\ P=10000 \\ -2P = -20000 \end{Bmatrix}$$

**9. Solution:** To solve this system, we eliminate the rows and columns corresponding to zero displacements, specifically 1, 2, 3, and 4, resulting in a reduced system:

$$\frac{EA}{L} \begin{bmatrix} 0.3536 & 0.3536 \\ 0.3536 & 1.3536 \end{bmatrix} \begin{Bmatrix} u_3 \\ v_3 \end{Bmatrix} = \begin{Bmatrix} 10000 \\ -20000 \end{Bmatrix}$$

After solving this system, we obtained the following results:

$$u_3 = 1.4654 \text{ mm}$$

$$v_3 = -0.7522 \text{ mm}$$

To calculate the reactions at the supports, we reconsider equations 1,2,3 and 4, substituting the displacements  $u_3$  and  $v_3$  with their previously determined values:

$$R_{x1} = -4 \times 10^4 \times 0.3536(u_3 + v_3) = -4 \times 10^4 \times 0.3536(1.4654 + (-0.7522)) = -10\,087.50 \text{ N}$$

$$R_{y1} = 4 \times 10^4 \times (-0.3536)u_3 + 4 \times 10^4 \times (-0.3536)v_3 = -10\,087.50 \text{ N}$$

$$R_{x2} = 0 \text{ N}$$

$$R_{y2} = 4 \times 10^4 \times (-1)v_3 = 4 \times 10^4 \times (-1) \times (-0.7522) = 30088 \text{ N}$$

After calculation, the results for the unknown displacements and reactions are summarized in the [Table 2.5:](#)

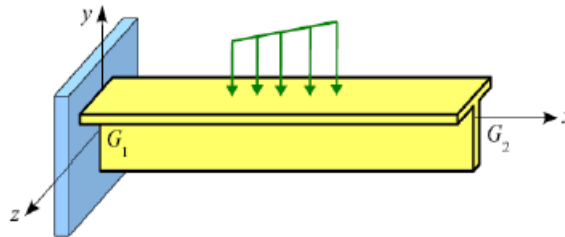
**Table 2.5:** Outcomes specific to the case study (truss structure under tensile loading)

Nodes	Displacements (u)	Displacements (v)	Forces /x-axis	Forces /y-axis
Node 1	$u_1 = 0$	$v_1 = 0$	-10 087.5 N	-10 087.5 N
Node 2	$u_2 = 0$	$v_2 = 0$	0 N	30 087.5 N
Node 3	$u_3 = 1.4654$ mm	$v_3 = -0.7522$ mm	10 000 N	-20 000 N

## 2.6 Beam elements

### 2.6.1 Definition of beam systems

A beam system typically refers to a structural framework composed of beams, which are long, straight members designed to support loads by resisting bending (Figure 2.9). In engineering and construction, beam systems are common components of buildings, bridges, and other structures where horizontal spans need to be supported. These systems are characterized by their ability to distribute loads primarily through bending moments and shear forces along their length. Beams are often categorized based on their cross-sectional shape, such as I-beams, H-beams, or rectangular beams, and are made from materials like steel, wood, or concrete, depending on the specific requirements of the structure. Beam systems are fundamental elements in structural analysis and design, and engineers utilize various analytical methods and techniques to ensure their stability, strength, and durability under different loading conditions [10].

**Figure 2.9:** Schematic representation of a beam system

### 2.6.2 Beam elements

Beam elements are usually one-dimensional (1D) and represent the longitudinal axis of the beam. They have two nodes at each element. These nodes define the endpoints of the element. The resistance of the beam to bending and axial deformation is represented by the stiffness matrix of a beam element. It takes into account the material properties of the beam, such as Young's modulus, cross-sectional area, and moment of inertia, as well as the element's geometry. Beam elements can be subjected to different boundary conditions, such as fixed supports, pinned supports or other types of restraints. The constraints applied to the nodes of the beam element are



determined by these boundary conditions. Beam elements can be subjected to different types of loads. These include concentrated loads, distributed loads, moments and thermal loads. Each of these types of load is applied at a specific point along the length of the beam element. In the FEM model, beam elements are combined with other types of finite elements to form a global stiffness matrix. The displacements, rotations and internal forces in the beam elements are determined by solving the system of equations representing the equilibrium of forces and moments.

### 2.6.3 Bending behavior of beam bar finite element

The analysis of the bending behavior of beam-bar finite elements is a common task in structural mechanics, allowing the prediction of how the structure will respond to applied loads, in terms of deformation, bending moments, and stresses. Computational tools like the finite element method facilitate this analysis for complex structures and varied loading conditions.

#### 2.6.3.1 Beam Theory

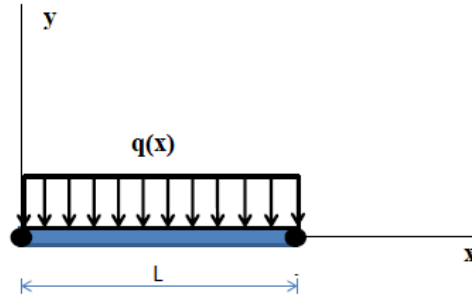
Beam theory provides essential tools for analyzing the behavior of beams under various loads. The Euler-Bernoulli theory is suitable for slender beams where shear deformation can be neglected, while the Timoshenko theory is more appropriate for thick beams where shear effects are significant. Understanding these theories is crucial for the design and analysis of safe and efficient structural systems. The basic concepts that form the foundation of beam theory are:

- **Beam Definition:** A beam is a long, slender structural member subjected primarily to loads perpendicular to its longitudinal axis;
- **Loads and Reactions:** Beams can be subjected to different types of loads such as point loads, distributed loads, and moments. Supports provide reactions that balance the applied loads, with common types being fixed, simply supported, and cantilever supports;
- **Shear Force ( $T$ ):** The internal force perpendicular to the longitudinal axis of the beam.
- **Bending Moment ( $M$ ):** The internal moment causing the beam to bend.
- **Normal Stress ( $\sigma$ ):** Induced by bending moments, varying linearly across the cross-section.
- **Shear Stress ( $\tau$ ):** Induced by shear forces, typically parabolic in distribution across the cross-section;

#### 2.6.3.2 Bending behavior

Let's consider a beam with a rectangular cross-section  $A$  and a length  $l$  subjected to a linearly varying bending load  $q(x)$  along the longitudinal axis  $x$  (Figure 2.10). We isolate an element  $dx$

of this beam as shown in the Figure 2.11. The beam bends under the load and undergoes a vertical displacement  $w(\mathbf{x})$  (deflection). Based on Bernoulli's theory, the movement of the beam in the  $(x,y)$  plane is described by the axial displacement  $u(\mathbf{x},y)$  and the vertical displacement  $w(\mathbf{x})$  as shown in the (Figure 2.11) [9].



**Figure 2.10:** Beam system under bending load  $q(x)$

Based on this Figure, the rotation  $\theta(\mathbf{x})$  of the deformed section is calculated by the following relation:

$$\theta = \text{tg}(\theta) = \frac{\partial w(x)}{\partial x} \quad (2.58)$$

The axial displacement  $u(\mathbf{x})$  induced by the rotation is given by the relation:

$$u(x) = -\theta y = -y \frac{dw(x)}{dx} \quad (2.59)$$

Hooke's law for an elastic medium allows the expression of the stress distribution along the cross-section of the beam by the following relations:

$$\sigma_x = E\varepsilon_x = E \frac{du}{dx} = -yE \frac{d^2w}{dx^2} \quad (2.60)$$

The basic principles of strength of materials allow expressing the bending moment of the beam by the following relation:

$$M(x) = -\int y \sigma_x dA = -EI \frac{d^2w}{dx^2} \quad (2.61)$$

With:

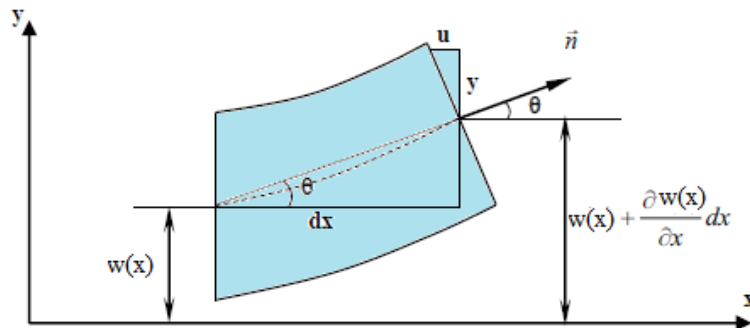
$$I = \int_A y^2 dA \quad (2.62)$$

Where  $M(x)$  is the bending moment at position  $x$ ,  $E$  is the Young's modulus,  $I$  is the moment of inertia, and  $w$  is the deflection of the beam. The uniformly distributed load  $q(x)$  can be expressed in terms of the shear force by the following relation:

$$q(x) = \frac{dT}{dx} \quad (2.63)$$

And the shear force by:

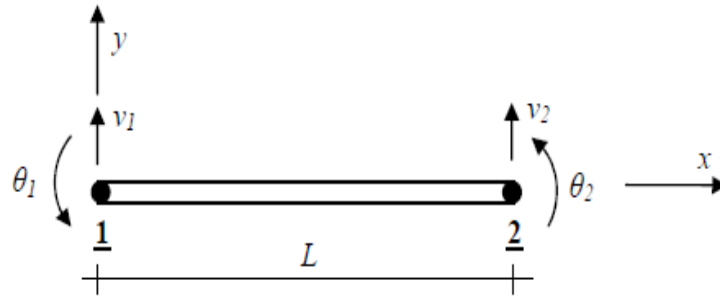
$$T(x) = \frac{dM(x)}{dx} = \frac{d}{dx} \left( EI \frac{d^2w}{dx^2} \right) \quad (2.64)$$



**Figure 2.11:** Representative beam element showing displacements and rotations due to bending load

#### 2.6.4 Formulation beam element

The formulation of a finite element beam involves developing mathematical models to represent the behavior of the beam under various loading conditions within the framework of finite element analysis. As shown in Figure 2.12, a beam element has two degrees of freedom at each node: a transverse displacement and a rotation [9].



**Figure 2.12:** Schematic representation of the degrees of freedom of a beam bar element

### 2.6.3.1 Stiffness matrix of a beam-bar element

This stiffness matrix relates the generalized deformations (displacements and rotations) to the generalized forces (axial forces and moments) for a beam-bar element subjected to axial and bending loads. In bending, each node of this element has two degrees of freedom: one in translation in the transverse direction ( $y$ -axis) denoted  $v$ , and one associated with the rotation  $\theta$  around the  $z$ -axis perpendicular to the  $(x,y)$  plane. Thus, four boundary conditions ( $v_1, \theta_1, v_2, \theta_2$ ) can be used to define its approximation function, which explains the expression of  $v(x)$  in the form of a polynomial of degree 3. The expression for the approximation function of the displacement function is given as follows:

$$w(x) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3 \quad (2.65)$$

Where,  $\alpha_i$  are the unknown coefficients of the polynomial chosen as the approximation function. The rotation is defined by the following relation:

$$\theta(x) = \frac{dw(x)}{dx} \quad (2.66)$$

If we substitute relation (2.65) into equation (2.66), we obtain:

$$\theta(x) = \alpha_2 + 2\alpha_3 x + 3\alpha_4 x^2 \quad (2.67)$$

The displacement vector is fully defined by combining the two relations (2.65) and (2.67).

$$\{u(x)\} = \begin{Bmatrix} w(x) \\ \theta(x) \end{Bmatrix} = \begin{bmatrix} 1 & x & x^2 & x^3 \\ 0 & 1 & 2x & 3x^2 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \quad (2.68)$$

In the compact form:

$$\{u(x)\} = [P(x)]\{\alpha\} \quad (2.69)$$

The application of the boundary conditions specific to this problem allows us to write:

$$\text{Node 1: } x=0 \quad w_1 = \alpha_1 \quad \theta_1 = \alpha_2$$

$$\text{Node 2: } X=L \quad w_2 = \alpha_1 + \alpha_2 L + \alpha_3 L^2 + \alpha_4 L^3 \quad \theta_2 = \alpha_2 + 2\alpha_3 L + 3\alpha_4 L^2$$

These equations are grouped and written in matrix form as follows:

$$\{u_n\} = \begin{Bmatrix} w_1 \\ \theta_2 \\ w_2 \\ \theta_2 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & L & L^2 & L^3 \\ 0 & 1 & 2L & 3L^2 \end{bmatrix} \begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} \quad (2.70)$$

The matrix A is denoted as follows:

$$[A] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & L & L^2 & L^3 \\ 0 & 1 & 2L & 3L^2 \end{bmatrix} \quad (2.71)$$

In a simplified matrix form, the system (2.70) is written as:

$$\{u_n\} = [A]\{\alpha\} \quad (2.72)$$

The inversion of this matrix leads to:

$$[A]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-3}{L^2} & \frac{-2}{L} & \frac{3}{L} & \frac{-1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & \frac{-2}{L^3} & \frac{1}{L^2} \end{bmatrix} \quad (2.73)$$

To determine the unknown coefficients  $\alpha_i$ , it suffices to invert the system (2.70), and we obtain:

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-3}{L^2} & \frac{-2}{L} & \frac{3}{L} & \frac{-1}{L} \\ \frac{2}{L^3} & \frac{1}{L^2} & \frac{-2}{L^3} & \frac{1}{L^2} \end{bmatrix} \begin{Bmatrix} w_1 \\ \theta_2 \\ w_2 \\ \theta_2 \end{Bmatrix} \quad (2.74)$$

Additionally, when represented in a more concise matrix format, equation (2.74) appears as:

$$\{\alpha\} = [A]^{-1} \{u_n\} \quad (2.75)$$

Substituting relation (2.75) into equation (2.69), we obtain:

$$\{u(x)\} = [P(x)][A]^{-1} \{u_n\} \quad (2.76)$$

According to elasticity theory, the strain field  $\boldsymbol{\varepsilon}(\mathbf{x})$  is deduced from the displacement field  $\mathbf{u}(\mathbf{x})$ , as follows:

$$\{\boldsymbol{\varepsilon}(\mathbf{x})\} = \frac{d}{dx}(\{\mathbf{u}(\mathbf{x})\}) = \frac{d}{dx}([\mathbf{P}(\mathbf{x})][\mathbf{A}]^{-1} \{\mathbf{u}_n\}) \quad (2.77)$$

Based on relations (2.60) and (2.61), we have:

$$\sigma_x = -yE \frac{d^2w}{dx^2}$$

And:

$$M(x) = -EI \frac{d^2w}{dx^2}$$

So:

$$\varepsilon(x) = \frac{\sigma(x)}{E} = -y \frac{d^2w}{dx^2} \quad (2.78)$$

The displacement approximation function  $w(\mathbf{x})$  given by equation (2.65) can be written in terms of the shape functions  $N_i(\mathbf{x})$  as follows:

$$w(\mathbf{x}) = \alpha_1 + \alpha_2 x + \alpha_3 x^2 + \alpha_4 x^3 = \sum_{i=0}^4 N_i(\mathbf{x}) \cdot u_i \quad (2.79)$$

The shape functions are determined based on the boundary conditions at the nodes:

$$\text{Node 1: for } x=1: \alpha_1 = w_1 \text{ and } \alpha_2 = \theta_1 \quad (2.80)$$

And:

$$\text{Node 2: for } x=L: w_2 = \alpha_1 + \alpha_2 L + \alpha_3 L^2 + \alpha_4 L^3 \text{ and } \theta_2 = \alpha_2 + 2\alpha_3 L + 3\alpha_4 L^2 \quad (2.81)$$

Substituting relation (2.80) into equation (2.81), we obtain:

$$\alpha_3 = -\frac{(2\theta_1 + \theta_2)}{L} - \frac{3(w_1 - w_2)}{L^2} \quad (2.82)$$

And:

$$\alpha_4 = -\frac{(\theta_1 + \theta_2)}{L^2} + \frac{2(w_1 - w_2)}{L^3} \quad (2.83)$$

Knowing, that the displacement vector is given by the following relation:

$$\{\mathbf{u}_i\}^T = [v_1 \quad \theta_1 \quad v_2 \quad \theta_2] \quad (2.84)$$

By substituting equations (2.80), (2.81), (2.82), (2.83) and (2.84) into equation (2.79) and then identifying, the expressions for the shape functions can be derived as follows:

$$N_1(x) = \frac{(2x^3 - 3Lx^2 + L^3)}{L^3} \quad (2.85)$$

$$N_2(x) = \frac{(x^3 - 2Lx^2 + L^2x)}{L^2} \quad (2.86)$$

$$N_3(x) = \frac{-(2x^3 - 3Lx^2)}{L^3} \quad (2.87)$$

$$N_4(x) = \frac{(x^3 - Lx^2)}{L^2} \quad (2.88)$$

$N_1(x)$ ,  $N_2(x)$ ,  $N_3(x)$  and  $N_4(x)$  represent the forms functions refer to the mathematical functions used to describe the shape of the beam element and how it deforms under load. In the context of a beam finite element, these functions are typically interpolation functions or shape functions. They describe how the displacement varies within the beam element, usually in terms of nodal displacements and help to approximate the displacement field and deformation behavior of the beam element. Once these shape functions are defined, the equation (2.79) becomes:

$$\begin{aligned} w(x) &= N_1(x)w_1 + N_2(x)\theta_1 + N_3(x)w_2 + N_4(x)\theta_2 \quad (2.89) \\ &= \frac{(2x^3 - 3Lx^2 + L^3)}{L^3}w_1 + \frac{(x^3 - 2Lx^2 + L^2x)}{L^2}\theta_1 - \frac{(2x^3 - 3Lx^2)}{L^3}w_2 + \frac{(x^3 - Lx^2)}{L^2}\theta_2 \end{aligned}$$

Substituting equations (2.89) into equation (2.78) leads to:

$$\varepsilon(x) = -y \frac{d^2}{dx^2} \begin{bmatrix} N_1(x) & N_2(x) & N_3(x) & N_4(x) \end{bmatrix} \begin{Bmatrix} w_1 \\ \theta_1 \\ w_2 \\ \theta_2 \end{Bmatrix} \quad (2.90)$$

After expanding this expression, we obtain:

$$\varepsilon(x) = -y \begin{bmatrix} \frac{d^2N_1(x)}{dx^2} & \frac{d^2N_2(x)}{dx^2} & \frac{d^2N_3(x)}{dx^2} & \frac{d^2N_4(x)}{dx^2} \end{bmatrix} \begin{Bmatrix} w_1 \\ \theta_1 \\ w_2 \\ \theta_2 \end{Bmatrix} \quad (2.91)$$

This expression reveals the matrix  $[B]$ , Such that:

$$[B] = -y \begin{bmatrix} \frac{d^2N_1(x)}{dx^2} & \frac{d^2N_2(x)}{dx^2} & \frac{d^2N_3(x)}{dx^2} & \frac{d^2N_4(x)}{dx^2} \end{bmatrix} \quad (2.92)$$

At this stage, and knowing the matrix  $[B]$ , we can determine the elemental stiffness matrix of the beam element defined by the following relation:

$$[K]^{(e)} = \int_V [B]^T [D] [B] dV \quad (2.93)$$

In this relation, the matrix  $[D]$  represents the stiffness of the beam. Substituting equation (2.92) into equation (2.93), we obtain:

$$[K]^{(e)} = E \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} y^2 dy dz \int_0^L \begin{bmatrix} \frac{d^2 N_1(x)}{dx^2} \\ \frac{d^2 N_2(x)}{dx^2} \\ \frac{d^2 N_3(x)}{dx^2} \\ \frac{d^2 N_4(x)}{dx^2} \end{bmatrix} \begin{bmatrix} \frac{d^2 N_1(x)}{dx^2} & \frac{d^2 N_2(x)}{dx^2} & \frac{d^2 N_3(x)}{dx^2} & \frac{d^2 N_4(x)}{dx^2} \end{bmatrix} dx \quad (2.94)$$

If we replace the second derivatives of the shape functions, the relation (2.89) becomes:

$$[K]^{(e)} = E \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} y^2 dy dz \int_0^L \begin{bmatrix} \frac{d^2 N_1(x)}{dx^2} \\ \frac{d^2 N_2(x)}{dx^2} \\ \frac{d^2 N_3(x)}{dx^2} \\ \frac{d^2 N_4(x)}{dx^2} \end{bmatrix} \begin{bmatrix} \frac{d^2 N_1(x)}{dx^2} & \frac{d^2 N_2(x)}{dx^2} & \frac{d^2 N_3(x)}{dx^2} & \frac{d^2 N_4(x)}{dx^2} \end{bmatrix} dx \quad (2.95)$$

Considering that:

$$E \int_{-\frac{h}{2}}^{\frac{h}{2}} \int_{-\frac{b}{2}}^{\frac{b}{2}} y^2 dy dz = EI \quad (2.96)$$

After evaluating the terms of this matrix product and integrating the different terms, we arrive at a final expression for the stiffness matrix of a beam element:

$$[K]^e = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \quad (2.97)$$

### 2.6.3.2 Vector force of a beam-bar element

The elemental force vector contains the components of the forces and moments applied to the nodes and on the boundaries of the element itself (concentrated forces and distributed loads). It is represented by a 1x4 column matrix. The first two values represent the forces and moments at the first node, while the last two represent those at the second node. As with the case of the two-node bar element, the distributed load is transformed into equivalent nodal loads (Figure 2.13). To

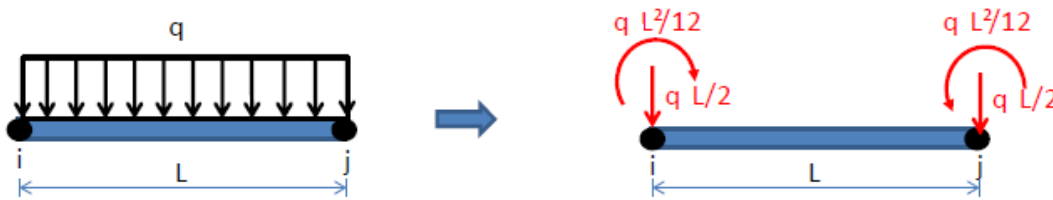


represent the vector force of a beam-bar element, we typically consider the forces and moments at both ends of the element. For a two-dimensional beam-bar element, the vector can be expressed as follows:

$$\{f\}^{(e)} = \int_L -q[N(x)]^T dx \quad (2.98)$$

If we substitute the values of the shape functions given by expressions (2.85) to (2.88) into relation (2.98), we obtain:

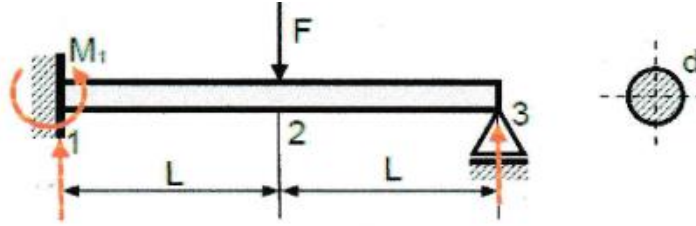
$$\{f\}^{(e)} = \int_L -q \begin{bmatrix} \frac{(2x^3 - 3Lx^2 + L^3)}{L^3} \\ \frac{(x^3 - 2Lx^2 + L^2x)}{L^2} \\ -\frac{(2x^3 - 3Lx^2)}{L^3} \\ \frac{(x^3 - Lx^2)}{L^2} \end{bmatrix} dx = \begin{Bmatrix} -q \frac{L}{2} \\ -q \frac{L^2}{12} \\ -q \frac{L}{2} \\ q \frac{L^2}{12} \end{Bmatrix} \quad (2.99)$$



**Figure 2.13:** Transformation of the distributed load into equivalent nodal loads

### 2.6.5 Example calculation of beam element

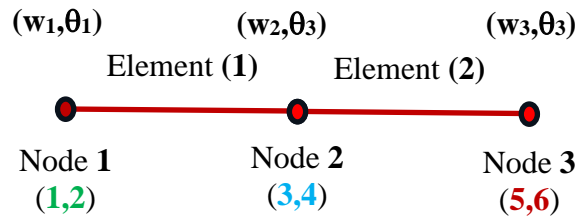
To illustrate the use of beam finite elements, we will examine a practical case study in this section. Our goal is to analyze the response of a beam structure under bending loading using the finite element method. The system consists of a circular section beam made of a material with Young's modulus  $E=2 \times 10^5$  MPa, with a diameter of  $d=60$  mm, fixed at one end and supported by a simple support at the other, with a length of  $2L=1.6$  m. The beam is subjected to a bending load  $F=4$  kN (Figure 2.14). We will analyze the system's response by determining the unknown displacements and the reactions at the supports.



**Figure 2.14:** Illustrates a demonstration of calculating the response

The procedure to address this issue involves the following steps:

- 1. Model of the beam structure:** The beam's geometric characteristics and material properties are depicted in [Figure 2.14](#).
- 2. Discretize of the beam:** Partition the beam into two finite elements between the nodes, and designate the nodes and elements accordingly. Nodes are labeled as 1, 2, 3, and elements as 1 and 2, as illustrated in [Figure 2.15](#).



**Figure 2.15:** Discretization of the beam into 2 linear elements and 3 nodes

- 3. Set up of the element stiffness matrices:** Compute the stiffness matrix for each element utilizing the formula specific to a beam element within the finite element method (FEM).

$$[K_e] = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

Let's calculate the flexural stiffness term:

$$I = \frac{\pi d^4}{64} = \frac{3.14 \times (60)^4}{64} = 635\,850 \text{ mm}^4$$

So:

$$\frac{EI}{L^3} = \frac{21 \times 10^4 \times 635\,850}{(800)^3} = 260.8 \frac{\text{N}}{\text{mm}}$$

- **Element 1:** between nodes 1 and 2, with cross-section A, Young's modulus E, and length

$$[K]^1 = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

- **Element 2:** between nodes 2 and 3. Since it has the same geometric and material properties as element 1, then:

$$[K]^2 = [K]^1 = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

**4. Assemble the global stiffness matrix:** Form the global stiffness matrix (GSM) by assembling the stiffness matrices of the individual elements.

dim (GSM) = number of degrees of freedom per node multiplied by the number of nodes

dim(GSM) = 2 × 3 = 6

$$[K] = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L & 0 & 0 \\ 6L & 4L^2 & -6L & 2L^2 & 0 & 0 \\ -12 & -6L & 24 & 0 & -12 & 6L \\ -6L & 2L^2 & 0 & 8L^2 & -6L & 2L^2 \\ 0 & 0 & -12 & -6L & 12 & -6L \\ 0 & 0 & 6L & 2L^2 & -6L & 4L^2 \end{bmatrix}$$

**5. Formulate the load vector:** Construct the global load vector {F} considering the applied concentrated force  $F$  at node 2.

- **Element 1:** between nodes 1 and 2, concentrated force at node 2:

$$\{\mathbf{f}\}^{(1)} = \begin{Bmatrix} F_1 \\ M_1 \\ F_2 \\ M_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ -\frac{F}{2} \\ 0 \end{Bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

- **Element 2:** between nodes 2 and 3, concentrated force at node 2:

$$\{\mathbf{f}\}^{(2)} = \begin{Bmatrix} F_2 \\ M_2 \\ F_3 \\ M_3 \end{Bmatrix} = \begin{Bmatrix} -\frac{F}{2} \\ 2 \\ 0 \\ 0 \end{Bmatrix} \begin{matrix} 2 \\ 3 \\ 4 \\ 5 \end{matrix}$$

**6. Global force vector:** The cumulative force vector represents the total forces and moments applied at the nodes throughout the entire finite element mesh. It is created by aggregating the contributions of all local force vectors, thus encapsulating all the forces and moments acting across the entire structure.

Dim (GVF) = number of degrees of freedom per node multiplied by the number of nodes

Dim (GVF) =  $2 \times 3 = 6$

$$\{\mathbf{F}\}^T = \{0 \quad 0 \quad -F \quad 0 \quad 0 \quad 0\}$$

**7. Applying boundary conditions:** The boundary conditions relevant to this case study are shown in [Table 2.6](#):

**Table 2.6:** The boundary conditions specific to the case study (beam under bending)

Nodes	Boundary conditions	Displacements	Forces
Node 1	fixed support	$w_1 = 0$ and $\theta_1 = 0$	$T_1 = R_1 = ?$ and $M_1 = ?$
Node 2	concentrated force	$w_2 = ?$ and $\theta_2 = ?$	$T_2 = -F$ and $M_2 = 0$
Node 3	simple support	$w_3 = 0$ and $\theta_3 = ?$	$T_3 = 0$ and $M_3 = 0$

**8. Formation of the global system:** To determine displacements and reactions at the supports (nodes 1 and 3), solve the system of equations using the relationship between the stiffness matrix, displacements, and external forces:  $[K] \cdot \{U\} = \{F\}$

$$260.8 \begin{bmatrix} 12 & 4800 & -12 & 4800 & 0 & 0 \\ 4800 & 256 \times 10^4 & -4800 & 128 \times 10^4 & 0 & 0 \\ -12 & -4800 & 24 & 0 & -12 & 4800 \\ 4800 & 128 \times 10^4 & 0 & 512 \times 10^4 & -4800 & 128 \times 10^4 \\ 0 & 0 & -12 & -4800 & 12 & -4800 \\ 0 & 0 & 4800 & 128 \times 10^4 & -4800 & 256 \times 10^4 \end{bmatrix} \cdot \begin{Bmatrix} w_1 = 0 \\ \theta_1 = 0 \\ w_2 = ? \\ \theta_2 = ? \\ w_3 = 0 \\ \theta_3 = ? \end{Bmatrix} = \begin{Bmatrix} T_1 = R_1 \\ M_1 = M_1 \\ T_2 = -4000 \\ M_2 = 0 \\ T_3 = R_3 \\ M_2 = 0 \end{Bmatrix}$$

**Solution:** To solve this system, we eliminate the rows and columns corresponding to nodes 1,2 and 5, which have zero displacements, resulting in the reduced system:

$$260.8 \begin{bmatrix} 24 & 0 & 4800 \\ 0 & 512 \times 10^4 & 128 \times 10^4 \\ 4800 & 128 \times 10^4 & 256 \times 10^4 \end{bmatrix} \cdot \begin{Bmatrix} w_2 \\ \theta_2 \\ \theta_3 \end{Bmatrix} = \begin{Bmatrix} -4000 \\ 0 \\ 0 \end{Bmatrix}$$

Finally, we find:

$$w_2 = -1.1183 \text{ mm}$$

$$\theta_2 = -0.000599 \text{ mm}$$

$$\theta_3 = -0.002396 \text{ mm}$$

To ascertain the reactions at the supports, we revisit equations 1,2 and 5, substituting the displacement  $w_2$  and rotations  $\theta_2$  and  $\theta_3$  with their previously calculated values.

$$R_1 = 260.8(-12w_2 + 4800\theta_2) = 260.8((-12) \times (-1.1183) + 4800 \times (-0.000599)) = 2750 \text{ N}$$

$$M_1 = 260.8(-4800w_2 + 128 \times 10^4 \theta_2) = 260.8((-4800) \times (-1.1183) + 128 \times 10^4 \times (-0.000599)) = 1200000 \text{ N.mm}$$

$$\begin{aligned} R_3 &= 260.8(-12w_2 - 4800\theta_2 - 4800\theta_3) \\ &= 260.8((-12) \times (-1.1183) + (-4800) \times (-0.000599) + (-4800) \times (-0.002396)) = 1250 \text{ N} \end{aligned}$$

Upon computation, the outcomes for the unknown displacements and reactions are presented in the following [Table 2.7](#):

**Table 2.7:** Results specific to the case study (beam under bending)

Nodes	Displacements		Loads	
	bend (mm)	rotation (rad)	Forces (N)	Moments (N.mm)
Node 1	$w_1 = 0$	$\theta_1 = 0$	$R_1 = 2750$	$M_1 = 1200000$
Node 2	$w_2 = -1.1183$	$\theta_2 = -0.000599$	$T_2 = F_2 = -4000$	$M_2 = 0$
Node 3	$w_3 = 0$	$\theta_3 = -0.002396$	$R_3 = 1250$	$M_3 = 0$

## 2.7 Conclusion

In this chapter, we explored the fundamental principles and applications of finite elements in structural analysis. We began with truss elements under tensile loading, where we examined how axial forces influence deformations and stresses in simple structures. Moving on to trusses, we studied structures composed of multiple bars. Finally, we investigated beam-bar elements, analyzing the response of beams under various bending loads. The combination of these concepts showcases the power of the finite element method in modeling and solving various engineering problems. By integrating different types of elements, we can analyze complex structures with high accuracy, predicting their behavior under various loading conditions. This ability to simulate and optimize structures is essential in modern design and engineering, providing robust and efficient solutions for a wide range of industrial applications.

## Chapter 3

# Programming and validation

### 3.1 Programming Languages

#### 3.1.1 Overview of programming languages

Programming languages are essential tools in coding, enabling developers to write instructions that computers can follow. They can be categorized into low-level and high-level languages. Low-level languages, such as machine language and assembly language, are closer to the computer's hardware and offer high performance, though they are more challenging to write and understand. High-level languages, such as Python, Java, C++, and JavaScript, are more abstract and user-friendly, making them easier to read, write, and maintain. These languages provide various features and libraries that help developers create complex software more efficiently [11].



**Figure 3.1:** popular programming languages

#### 3.1.2 Core Concepts in Programming

Key concepts in programming include variables, data types, control structures, functions, and objects. Variables are storage locations identified by names that hold data, which can be manipulated throughout the program. Data types define the kind of data a variable can hold, such

as integers, floats, strings, and Booleans. Control structures, like loops and conditionals, direct the flow of the program by executing different code blocks based on certain conditions. Functions or methods are reusable blocks of code designed to perform specific tasks and can take inputs (parameters) and return outputs. In object-oriented programming (OOP), objects and classes are fundamental. Objects are instances of classes, which define the structure and behavior of these objects, promoting code reuse and modularity [12].

### ***3.1.3 Development Paradigms***

Programming follows various development paradigms, including procedural programming, object-oriented programming, and functional programming. Procedural programming focuses on a sequence of instructions to perform tasks, emphasizing linear execution. Object-oriented programming organizes code into objects that encapsulate data and methods, enhancing modularity and code reuse. Functional programming, on the other hand, treats computation as the evaluation of mathematical functions and avoids changing state and mutable data [13].

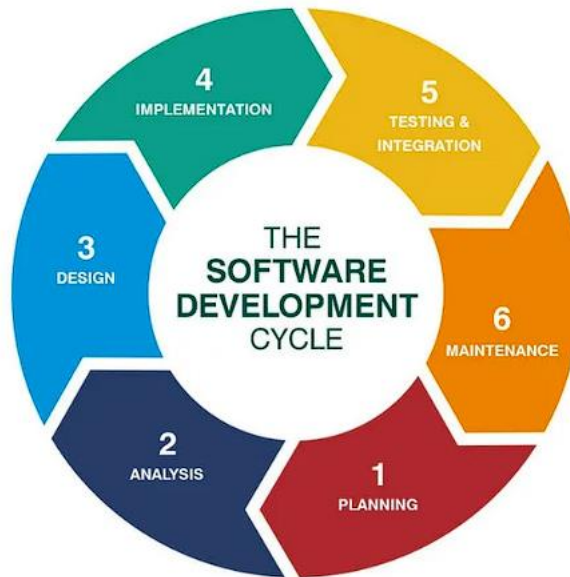
### ***3.1.4 Development Tools***

Development tools play a crucial role in programming, providing environments where code can be written, tested, and debugged. Integrated Development Environments (IDEs) like Visual Studio Code, PyCharm, and Eclipse offer comprehensive features that streamline the development process. Version control systems such as Git and SVN help manage changes to the source code over time, facilitating collaboration among developers [14].

### ***3.1.5 Software Development Lifecycle***

The software development lifecycle comprises several stages: planning, design, implementation, testing, deployment, and maintenance. Planning involves defining the scope and purpose of the software, while design focuses on creating blueprints for the software architecture. Implementation is the actual coding phase, followed by testing to ensure the software is bug-free and performs as expected. Deployment releases the software to users, and maintenance involves updating and fixing the software post-release to address any issues or new requirements [15].





**Figure 3.2:** Software Development Lifecycle

### 3.1.6 Popular Programming Languages and Their Uses

Various programming languages have become popular due to their unique strengths and use cases. Python is known for its simplicity and versatility, making it popular for web development, data analysis, artificial intelligence, and scientific computing. Java is commonly used in enterprise environments, Android app development, and large systems due to its robustness and portability. JavaScript is essential for web development, both on the client-side and server-side, thanks to its ability to create interactive and dynamic web pages. C++ is favored in system and software development, game development, and performance-critical applications for its efficiency and control over system resources. Ruby, with its simplicity and productivity, is often used in web development with the Ruby on Rails framework [16].

### 3.1.7 Emerging Trends

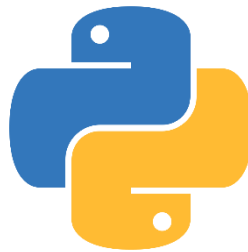
Emerging trends in programming include machine learning and artificial intelligence, which are increasingly integrated into applications to provide intelligent features and automation. Quantum computing, although still in its early stages, promises to revolutionize problem-solving with new programming paradigms, block chain technology is gaining traction for developing decentralized applications and crypto currencies, while the Internet of Things (IoT) focuses on programming interconnected devices and sensors to create smart environments.

Programming is a dynamic and ever-evolving field that adapts to technological advancements and changing user needs. Mastering programming languages and understanding core concepts are essential for developing efficient, innovative software solutions [17].

## 3.2 Chosen Programming Language (Python)

### 3.2.1 Definition and history of python

Python is a cross-platform, multi-paradigm, object-oriented programming language (Figure 3.3). It supports structured, functional and object-oriented imperative programming and is comparable to other languages in the same programming paradigm, such as Perl, Ruby, Scheme, Smalltalk and Tcl.



**Figure 3.3:** python logo

The language is dynamic, which allows for rapid development and flexibility. Automatic memory management using garbage collection is another feature which makes it more efficient and easier to maintain. Python also includes a comprehensive exception handling system which provides a robust solution for handling errors in a structured way.

Furthermore, educationalists have identified the language as an accessible introduction to the fundamental concepts of programming, as its syntax is clearly separated from low-level mechanisms.

Python is a versatile programming language that can be employed in a multitude of contexts and adapted to a wide range of applications through the use of specialized libraries. It is particularly popular as a scripting language for automating simple yet time-consuming tasks, such as retrieving the weather forecast from the Internet or integrating it into computer-aided design software to automate repetitive sequences of actions.

Computer-aided design software is one area where Python is used to automate repetitive sequences of actions (see the Adoption section). Additionally, it is employed as a prototype development language when a functional application is required prior to optimization with a

lower-level language. It is particularly prevalent in the scientific community and boasts numerous extensions for numerical applications.

The Python language is licensed under an open-source license similar to the BSD licence<sup>3</sup> and runs on a wide range of computer platforms, including supercomputers, mainframes<sup>4</sup>, Windows, Unix, GNU/Linux, Mac OS, Android and iOS, as well as Java and .NET.

The language is designed to optimize the productivity of programmers by offering high-level tools and an easy-to-use syntax [18].

### 3.2.2 *Python capabilities and functions*

Python offers a diverse set of capabilities that cater to various programming needs:

- Writing small, very simple programs, known as scripts, which perform specific tasks on your computer.
- Developing complete programs, such as games, office suites, multimedia software, e-mail clients, and more.
- Handling very complex projects, including software packages a collection of multiple pieces of software that can work together, commonly utilized in professional settings.
- Creating graphical interfaces to enhance user interaction and experience.
- Facilitating information circulation across a network, enabling seamless communication between devices.
- Engaging in advanced dialogue with your operating system, allowing for efficient system-level operations and automation.

### 3.2.3 *Errors and exceptions*

The process of programming is inherently complex, as is any human activity. Errors in programming are referred to as "bugs," and the techniques used to detect and correct them are collectively known as "debugging." It is evident that the most crucial skills to be acquired during the learning process are those required to debug a program effectively.

To gain a deeper understanding of the context of our work, it is first necessary to elucidate a few related concepts.

**A) Types of Programming Error:** As is the case with all programming languages, three types of error can occur in a Python program:

- Syntax errors: Python can only execute a program if the syntax is perfectly correct. If this is not the case, the process is interrupted and an error message is displayed (Figure.3.4). The

term syntax refers to the language syntax rules that the authors have set up for the structure of the program.

- The second type of error is the semantic error or logical error. If there is an error of this type in one of your programs, it functions correctly in the sense that no error message is generated. However, the result is not what was intended. The program instructions in the sequence do not correspond to the desired objective. The semantics (logic) are incorrect.
- The third category of error is the execution error, which manifests when a program is already operational but under specific conditions. For instance, if a program attempts to read a file that no longer exists, an execution error will occur. These errors are also known as exceptions because they typically indicate that an unforeseen event has occurred.

**B) The distinction between an error and an exception:** In general, an error is a response to a problem posed by the system. However, exceptions are alerts whose location and behavior are defined by the developer. Exceptions are used to handle errors; the reverse makes no sense.

**C) Error and exception handling:** Error handling is usually solved by recording the state of execution at the time of the error and interrupting the normal flow of the program to execute a special function or piece of code, which is known as the exception handler. Depending on the type of error ("division by zero", "file open error" and so on) that had occurred, the error handler can resolve the problem and the program can then continue with the data previously saved.

Exception handling is a construct in some programming languages for automatically handling or dealing with errors. Many programming languages such as C++, Objective-C, PHP, Java, Ruby, Python and many others have built-in support for exception handling.

When an exception occurs the normal flow of the program is disrupted and the program/application terminates abnormally, which is not recommended, therefore these exceptions must be handled.

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-1-2944c084f5c5> in <cell line: 1>()
----> 1 UB = U[B]
      2 UB
      3 PA = P[A]
      4 PA
      5
NameError: name 'U' is not defined
```

**Figure 3.4:** Undefined variable error example in python

Unfortunately, using the built-in support for exception handling requires exception handling needs for a deep beginner's knowledge of how these exceptions occur. In addition, full details can be given by the compiler to guide and help learners catch these exceptions. When a learner's program crashes at runtime, the learner receives feedback that an exception has occurred. However, the cognitive level of a beginner cannot allow him to catch these exceptions to avoid possible errors. Python (version 3.4) has around 29 built-in exceptions ([Table 3.1](#)) [19].

**Table 3.1:** Some standard Python built-in exceptions

Exception name	Description
Arithmetic Error	Basic exception for all errors that occurs numerical calculation.
Floating Point Error	Triggered when a floating point calculation fails.
Zero Divison Error	Triggered when division or modulo by zero occurs for all numeric types.
EOF Error	Triggered when there is no input from either <code>raw_input ()</code> or <code>input ()</code> and the end of the file is reached.
Index Error	Triggered when an index is not in a sequence.
IO Error	The event is triggered when an input/output operation fails, such as the print statement or the <code>open ()</code> function when an attempt is made to open a file that does not exist.
OS Error	For operation system errors
Value Error	In the event that the integrated function for a data type is of the correct type for the arguments, yet the arguments themselves are of an invalid value.

### 3.2.4 Jupyter notebook

We used Jupyter's notebook for easy troubleshooting and debugging and to well organize our code ([Figure 3.5](#)).



**Figure 3.5:** Jupyter Logo

### 3.2.5 Overview of Jupyter notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It's a powerful tool

widely used in data science, scientific computing, and machine learning for interactive computing [20].

### 3.2.6 Key features of Jupyter notebook

- Interactive Code Execution
- Rich Text Support
- Data Visualization
- Documentation and Sharing
- Extensibility

### 3.3 Finite element program

In order to achieve the desired results and guarantee a well-built program we need to respect the geometric and physical description of the problem studied the calculation of the elementary matrices and vectors and assembly of the overall system, solving the system and finally visualizing the results. The actual use of finite element modeling software on the market provides certain know-how and expertise adapted to the problem being addressed, a good knowledge of the physics involved and a general understanding of the finite element method.

#### 3.3.1 Finite element program execution steps

Finite Element Analysis (FEA) involves several methodical steps to accurately model and analyze physical systems (Figure.3.6). Each one of the previous steps contain a number of conditions and elements that is related to how the program function, which are and not limited to:

Step 1: input

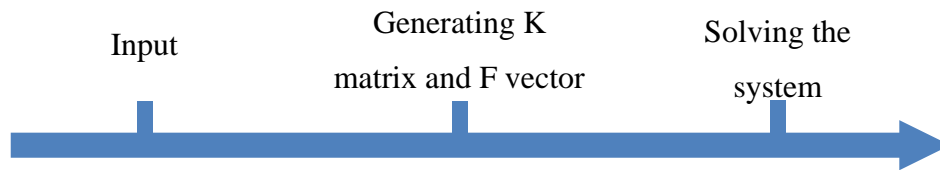
Consists of the different variables and parameters the program needs to function like each node coordination and how elements are linked together, physical parameters and demands as well as limit conditions.

Step 2: Generation of K matrix and F vector

Starting with extracting each information related to an element then building the global matrix K and vector F and finally the Assembly process.

Step 3: Solving the system

Applying: the limit conditions on the matrix and vector, Gaussian elimination and calculating U



**Figure 3.6:** program execution steps

### 3.3.2 Developed program

Finite element programs are available in the net on variety of programming languages which most are in MATLAB, however there are nearly none in python therefore the method can be easily exported as an API.

### 3.3.3 Code

Out of the numerous systems we addressed three; 1 Dimensional bar, Trillis and Beam. All of them share the same libraries as follow:

- NumPy: library is used for numerical computing in Python, widely used for its powerful capabilities in handling and manipulating numerical data.
- Pandas: Pandas is a powerful and flexible open-source data analysis and manipulation library for Python. It is built on top of NumPy and provides high-level data structures and functions designed to make data analysis fast and easy.
- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots and graphs and is an essential tool in the data analysis and data science toolkit.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

**Figure 3.7:** Used python libraires

### 3.3.4 Bar program

The bar program will perform a linear static analysis on a bar subjected to axial loads. The bar will be divided into elements, and the program will calculate the displacements at the nodes and the stresses in each element. The structured Python program that performs this analysis is given below:

**Phase 1: input**

This is the input part of nodes and boundary conditions, where data is an array that stores “x” and “y” as coordinates, “U” as displacement and “F” as the load of each node (Figure.3.8).

```
columns = ['coordinate_x', 'coordinate_y', 'displacement_x', 'load_x']
data = [[x, y, U, F]]
nodes = pd.DataFrame(data, columns = columns)
nodes
```

**Figure 3.8:** Bar nodes parameters

This is the input part of elements where we define the 2 nodes that represent each single element (Figure.3.9).

Note: Unknown variables are described as “np.nan”.

```
columns = ['start', 'end', 'area', 'material']
data = [[starting_node, ending_node, A, E]]
elements = pd.DataFrame(data, columns = columns)
elements
```

**Figure 3.9:** Bar element parameters

- p.DataFrame is a function in pandas library that has a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

**Phase 2: elementary stiffness matrix**

The elementary stiffness matrix represents the stiffness of a single finite element within the larger system and is crucial for assembling the global stiffness matrix. For a 1D bar element subjected to axial loads, the elementary stiffness matrix can be derived from the governing equations of linear elasticity. For a bar element with length  $L_e$ , cross-sectional area  $A$ , and Young's modulus  $E$ , the program for computing the elementary stiffness matrix  $[k]$  for a bar element is given below (Figure.3.10):



```

defcompute(element):
    start = element['start']
    end = element['end']
    x_coordinate_start = nodes.loc[start, 'coordinate_x']
    y_coordinate_start = nodes.loc[start, 'coordinate_y']
    x_coordinate_end = nodes.loc[end, 'coordinate_x']
    y_coordinate_end = nodes.loc[end, 'coordinate_y']
    deltaX = x_coordinate_end - x_coordinate_start
    deltaY = y_coordinate_end - y_coordinate_start
    length = np.sqrt(deltaX ** 2 + deltaY ** 2)
    stiffness = element['material'] * element['area'] / length
    R = np.array([[1, -1],
                 [-1, 1]])
    return length, stiffness, R
elements[['length', 'stiffness', 'R']] = elements.apply(compute, axis =
1, result_type='expand')
elements

```

**Figure 3.10:** Bar elementary stiffness matrices calculation

- .loc function is a property that allows Pandas to query data within a data frame in a standard format.
- .array is a linear data structure where all elements are arranged sequentially.

### Phase 3: Stiffness matrix Global

This phase assembles the elementary stiffness matrices into a global one (Figure.3.11).

```

defcompute_globalK(element):
    N = len(nodes)
    indices = np.arange(N)
    indices = indices.reshape(-1,1)
    K = np.zeros((N, N))
    start = element['start']
    end = element['end']
    indices = np.hstack([indices[start], indices[end]])
    K[np.ix_(indices, indices)] = element['stiffness'] * element['R']
    return K
K = elements.apply(compute_globalK, axis=1).sum()
K.round(4)

```

**Figure 3.11:** Bar global matrix calculation

- len returns the length of given variable.
- .arrange allows you to create arrays with evenly spaced values.

- `.hstack` function is used to horizontally stack data of the variables.

#### Phase 4: Gaussian elimination

Where we shrink the stiffness matrix global into a smaller one for further calculation

```
defpartition_K(K, A, B):
    KAA = K[np.ix_(A, A)]
    KAB = K[np.ix_(A, B)]
    KBA = K[np.ix_(B, A)]
    KBB = K[np.ix_(B, B)]
    return KAA, KAB, KBA, KBB
U = nodes[['displacement_x']].to_numpy()
U = U.ravel()
A = np.isnan(U)
P = nodes[['load_x']].to_numpy()
P = P.ravel()
B = np.isnan(P)
KAA, KAB, KBA, KBB = partition_K(K, A, B)
KAA
```

**Figure 3.12:** Bar Gaussian elimination process

- `.ravel` is used to change a 2-dimensional array or a multi-dimensional array into a contiguous flattened array.
- `.isnan` returns if given values are not a void.

#### Phase 5: Results

Calculates the final results “Unknown displacement” and “Unknown loads” ” (Figure.3.13).

```
UB = U[B]
UB
PA = P[A]
PA
UA = np.dot(np.linalg.inv(KAA), (PA - np.dot(KAB,UB)))
U[A] = UA
PB = np.dot(KBA, UA) + np.dot(KBB, UB)
P[B] = PB
result = nodes.copy()
result[['displacement_x']] = U.reshape(-1,1)
result[['load_x']] = P.reshape(-1,1)
result
```

**Figure 3.13:** Bar results

- `.linalg` defines linear algebra functions.
- `.inv` inverses a matrix.

### 3.3.5 Truss program

Truss program is similar to the previous program therefore we decided to show only the difference between the two.

#### Phase 1: input

In Input phase we added only the displacement “Uy” and load “Fy” as follow (Figure.3.14):

```
columns =
['coordinate_x', 'coordinate_y', 'displacement_x', 'displacement_y', 'load_x', 'load_y']
data = [[x, y, Ux, Uy, Fx, Fy]]
nodes = pd.DataFrame(data, columns = columns)
nodes
```

**Figure 3.14:** Truss node parameters

Element part is still the same as (Figure 3.9).

#### Phase 2: elementary stiffness matrix

The difference here is the elementary stiffness matrix components (Figure.3.15).

```
defcompute(element):
    start = element['start']
    end = element['end']
    x_coordinate_start = nodes.loc[start, 'coordinate_x']
    y_coordinate_start = nodes.loc[start, 'coordinate_y']
    x_coordinate_end = nodes.loc[end, 'coordinate_x']
    y_coordinate_end = nodes.loc[end, 'coordinate_y']
    deltaX = x_coordinate_end - x_coordinate_start
    deltaY = y_coordinate_end - y_coordinate_start
    length = np.sqrt(deltaX ** 2 + deltaY ** 2)
    stiffness = element['material'] * element['area'] / length
    c = deltaX / length
    s = deltaY / length
    R = np.array([[c*c, c*s, -c*c, -c*s],
                  [c*s, s*s, -c*s, -s*s],
                  [-c*c, -c*s, c*c, c*s],
                  [-c*s, -s*s, c*s, s*s]])
    return length, stiffness, R
elements[['length', 'stiffness', 'R']] = elements.apply(compute, axis =
1, result_type='expand')
elements
```

**Figure 3.15:** Truss elementary stiffness matrices calculation

#### Phase 3: stiffness matrix global

Assembly is the same regardless of the system

### Phase 4: Gaussian elimination

In the Gaussian elimination phase we added displacement y “Uy” and load y “Fy” (Figure.3.16).

```
defpartition_K(K, A, B):
    KAA = K[np.ix_(A, A)]
    KAB = K[np.ix_(A, B)]
    KBA = K[np.ix_(B, A)]
    KBB = K[np.ix_(B, B)]
    return KAA, KAB, KBA, KBB
# index A is where the displacement is unknown
U = nodes[['displacement_x', 'displacement_y']].to_numpy()
U = U.ravel()
A = np.isnan(U)
P = nodes[['load_x', 'load_y']].to_numpy()
P = P.ravel()
B = np.isnan(P)
KAA, KAB, KBA, KBB = partition_K(K, A, B)
KAA
```

**Figure 3.16:** Truss Gaussian elimination process

### Phase 5: results

Same as phase 1 and 4 we added displacement y “Uy” and load y “Fy” (Figure.3.17).

```
UB = U[B]
UB
PA = P[A]
PA
UA = np.dot(np.linalg.inv(KAA), (PA - np.dot(KAB, UB)))
U[A] = UA
PB = np.dot(KBA, UA) + np.dot(KBB, UB)
P[B] = PB
result = nodes.copy()
result[['displacement_x', 'displacement_y']] = U.reshape(-1, 2)
result[['load_x', 'load_y']] = P.reshape(-1, 2)
result
```

**Figure 3.17:** Truss results

### 3.3.6 Beam program:

Truss program is similar to the previous program therefore we decided to show only the difference between the two.

**Phase 1: input**

In Input phase got only one displacement  $y$  “W” and we added a rotation “ $\theta$ ” as well as torque “M” (Figure.3.18).

```
columns = ['coordinate_x', 'coordinate_y', 'displacement_y', 'rotation', 'load_y', 'torque']
data = [[x, y, W,  $\theta$ , T, M]]
nodes = pd.DataFrame(data, columns = columns)
nodes
```

**Figure 3.18:** Beam node parameters

The element part changed as follow: we replaced area with width “b” and height “h” (Figure.3.19).

```
columns = ['start', 'end', 'inertia', 'material', 'length']
data = [[starting node, ending node, I, E, L]]
elements = pd.DataFrame(data, columns = columns)
elements
```

**Figure 3.19:** Beam element parameters**Phase 2: elementary stiffness matrix**

Stiffness law changed and elementary stiffness matrix as well (Figure.3.20).

```
def compute(element):
    start = element['start']
    end = element['end']
    x_coordinate_start = nodes.loc[start, 'coordinate_x']
    y_coordinate_start = nodes.loc[start, 'coordinate_y']
    x_coordinate_end = nodes.loc[end, 'coordinate_x']
    y_coordinate_end = nodes.loc[end, 'coordinate_y']
    stiffness=(element['material']*element['inertia'])/(element['length']*element
['length' ]*element['length' ])
    L = element['length' ]
    R= np.array([[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]])

    return stiffness, R
elements [['stiffness', 'R']] = elements.apply(compute, axis = 1, result_type='expand')
elements
```

**Figure 3.20:** Beam elementary stiffness matrices calculation**Phase 3: stiffness matrix global**

Assembly is the same regardless of the system

**Phase 4: Gaussian elimination**

In this phase displacement and load got replaced by rotation “ $\theta$ ” and torque “M” (Figure.3.21).

```

def partition_K(K, A, B):
    KAA = K[np.ix_(A, A)]
    KAB = K[np.ix_(A, B)]
    KBA = K[np.ix_(B, A)]
    KBB = K[np.ix_(B, B)]
    return KAA, KAB, KBA, KBB

# index A is where the displacement is unknown
U = nodes[['displacement_y', 'rotation']].to_numpy()
U = U.ravel()
A = np.isnan(U)
P = nodes[['torque', 'load_y']].to_numpy()
P = P.ravel()
B = np.isnan(P)
KAA, KAB, KBA, KBB = partition_K(K, A, B)
KAA

```

**Figure 3.21:** Beam Gaussian elimination process

### Phase 5: results

Same as phase 1 and 4 we replaced displacement  $x$  and load  $x$  with rotation “ $\theta$ ” and torque “ $M$ ” (Figure.3.22).

```

UB = U[B]
UB
PA = P[A]
PA
UA = np.dot(np.linalg.inv(KAA), (PA - np.dot(KAB, UB)))
U[A] = UA
PB = np.dot(KBA, UA) + np.dot(KBB, UB)
P[B] = PB
result = nodes.copy()
result[['displacement_y', 'rotation']] = U.reshape(-1, 2)
result[['load_y', 'torque']] = P.reshape(-1, 2)
result

```

**Figure 3.22:** Beam results

## 3.4 Validating results using RMD6 software

### 3.4.1 Overview of RDM6

RDM6 or "Resistance des Matériaux 6" is software used in mechanical and civil engineering for analyzing the strength and resistance of materials. It is commonly used for educational

purposes, and provides tools to solve problems related to material resistance and structural analysis. It's a software tool designed for the simulation and analysis of material strength, intended for engineers and researchers working in the field of structural mechanics. This software allows users to model, simulate, and analyze the behavior of materials under various constraints and load conditions [21].

### 3.4.2 RDM6 Features



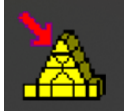

Rdm6 offers a range of features to help engineers of which are [21]:

- Material resistance calculation; including stress, strains and deformation.
- Structural analysis; provides tools for analyzing beams, columns, and other structural elements to determine their behavior under various loading conditions.
- Educational tools; often used in engineering courses to help students understand and apply principles of material resistance and structural mechanics.
- Applications; that are used to design and evaluate the safety and performance of buildings, bridges, and other structures.

### 3.4.3 RDM6 Modules

The RDM6 structural analysis software is composed of various modules that enable comprehensive and detailed analyses of material and structural behaviors under different conditions. It contains 4 modules or sub-software; Flexion, Ossatures, Element Finis and Rosette. Below is an overview of the main modules available in the RDM6 software (Table 3.2) [21].

**Table 3.2:** The main modules available in the RDM6

<b>Flexion</b>	Flexion is software designed for static analysis, using the finite element method, of straight beams subjected to simple bending.	
<b>Ossatures</b>	This software allows the study, using the finite element method, of the static and dynamic behavior of frameworks.	
<b>Element Finis</b>	The evaluation of the mechanical and/or thermal behavior of a part using the finite element method.	
<b>Rosette</b>	Analysis of strain gauge rosettes, study of stresses and deformations around a point.	

### 3.5 Comparing program results with traditional analytical results

In the field of structural analysis and material strength, it is crucial to validate the accuracy and reliability of computational tools. This section focuses on comparing the results obtained

from the developed programs with those derived from manual calculations in simple cases and with those provided by RDM 6 software in more complex cases. Such comparisons are essential to ensure that the developed programs can replicate established theoretical predictions. By examining the consistency between program results, manual calculations, and RDM 6 software outputs, we can assess the software's performance, identify any discrepancies, and understand the underlying reasons for these differences. This process not only enhances confidence in the software's capabilities but also contributes to its continuous improvement and refinement.

### 3.5.1 Example of the calculation for bar element in tension

For the first example of the calculation of the bar element in tension (Paragraph 2.4.4), the comparison is between the manual results of the finite element method (Table 3.3) and those obtained by the developed program for the 1D bar element in tension (Table 3.4).

**Table 3.3:** Outcomes provided by manual computing for bar element

Nodes	Displacements (mm)	Forces (N)	
Node 1	$u_1 = 0$	$F_1 = 100$	$R_1 = -152.94$
Node 2	$u_2 = 0.0026$	$F_2 = 100$	
Node 3	$u_3 = 0.0021$	$F_3 = 200$	
Node 4	$u_4 = 0$	$R_4 = -247.06$	

**Table 3.4:** Results derived from proposed program for bar element

	coordinate_x	coordinate_y	displacement_x	load_x
0	0	0	0.0000	-153
1	2000	0	0.0026	100
2	3000	0	0.0021	200
3	4000	0	0.0000	-248

To compare these results, the relative error calculated using the following formula (3.1) is shown in Table 3.5.

$$e\% = \frac{|y_{\text{program}} - y_{\text{manual}}|}{y_{\text{manual}}} \times 100 \quad (3.1)$$



**Table 3.5:** Relative error between outcomes provided by manual computing and those from proposed program for bar element

Nodes	(e %) displacements	(e %) Forces	
Node 1	0	0	0.04
Node 2	0	0	
Node 3	0	0	
Node 4	0	0.4	

Table 3.5 presents the relative errors between manual calculation's results and the proposed program's results for a bar element consisting of 3 elements and 4 nodes; we observe that on displacement's section there is no errors, on force's section there is error valued with 0.04% in node 1, in node 4 there is valued with 0.4%, if we gather all the errors we have a total error valued with 0.44%, the total error is under 1% so we can say that the solution of the proposed program is almost same as manual calculation in bar elements.

### 3.5.2 Example of the calculation for truss element

In the second example involving the calculation of the truss element in tension (Paragraph 2.5.5), the comparison is made between the manual finite element method results (Table 3.6) and those obtained using the developed program for the truss element in tension (Table 3.7).

**Table 3.6:** Results obtained through manual calculations for truss element

Nodes	Displacements (u)	Displacements (v)	Forces /x-axis	Forces /y-axis
Node 1	$u_1 = 0$	$v_1 = 0$	-10 087.5 N	-10 087.5 N
Node 2	$u_2 = 0$	$v_2 = 0$	0 N	30 087.5 N
Node 3	$u_3 = 1.4654 \text{ mm}$	$v_3 = -0.7522 \text{ mm}$	10 000 N	-20 000 N

**Table 3.7:** Results derived from the proposed program for truss element

coordinate_x	coordinate_y	displacement_x	displacement_y	load_x	load_y
0	0	0	0.0000	0.0000	-10000 -10175
1	100	0	0.0000	0.0000	0 30000
2	100	50	1.4654	-0.7522	10000 -20000

To contrast these findings, the relative discrepancy computed via formula (3.1) is displayed in Table 3.8.

**Table 3.8:** Relative error between outcomes provided by manual computing and those from proposed program for truss element

Nodes	(e%) u	(e%) v	(e %) F /x-axis	F /y-axis
Node 1	0	0	0.87	0.87
Node 2	0	0	0	0.87
Node 3	0	0	0	0

Table 3.8 presents the relative errors between manual calculation's results and the proposed program's results for a truss consisting of 3 bars and 3 nodes; we observe that on displacement's section there is no errors, on force's section there is error valued with 0.87% in node 1 on F /x-axis and F /y-axis, in node 2 there is on F /y-axis with the same value, if we gather all the errors we have a total error valued with 2.61%, the total error is under 5% so we notice that the proposed program is solving the system with a reasonable solution comparing to manual calculation in simple truss structures.

### 3.5.3 Example of the calculation for a bar element in bending

In this instance, concerning the calculation of the bar element in bending (Paragraph 2.6.5), the comparison extends to the manual finite element method results (Table 3.9) and those derived from the developed program for the bar element in tension (Table 3.10).

**Table 3.9:** Results derived from manual calculations for beam element

Nodes	Displacements		Loads	
	bend (mm)	rotation (rad)	Forces (N)	Moments (N.mm)
Node 1	$w_1 = 0$	$\theta_1 = 0$	$R_1 = 2750$	$M_1 = 1200000$
Node 2	$w_2 = -1.1183$	$\theta_2 = -0.000599$	$T_2 = F_2 = -4000$	$M_2 = 0$
Node 3	$w_3 = 0$	$\theta_3 = -0.002396$	$R_3 = 1250$	$M_3 = 0$

**Table 3.10:** Results derived from the proposed program for beam element

	coordinate_x	coordinate_y	displacement_y	rotation	load_y	torque
0	0	0	0.0000	0.0000	2730	1200000
1	800	0	-1.1178	-0.0006	-4000	0
2	1600	0	0.0000	-0.0024	1270	0

To compare these results, Table 3.11 illustrates the relative error calculated using formula (3.1) between the outcomes generated by manual computation and those obtained from the proposed program for the beam element.

**Table 3.11:** Relative error between the outcomes obtained through manual computation and those generated by the proposed program for the beam element

Nodes	(e%) displacements		(e%) loads	
	bend	rotation	Forces	Moments
Node 1	0	0	0.72	0
Node 2	0.045	0.17	0	0
Node 3	0	0.17	0.72	0

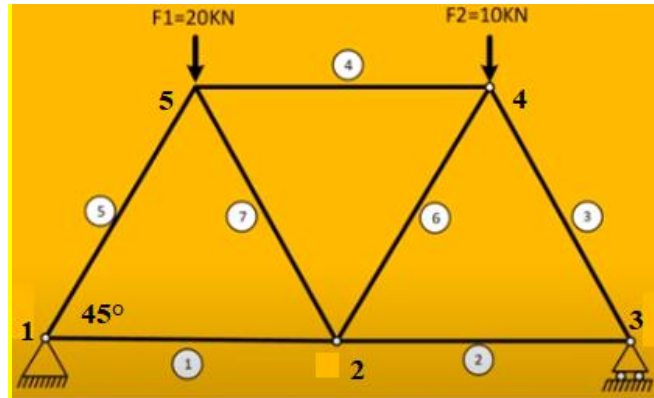
Table 3.11 presents the relative errors between manual calculation's results and the proposed program's results for a beam contains 2 elements and 3 nodes with a circular section; we observe that in node 2 on bend there is an error that valued with 0.045%, on rotation the error is 0.17%, in node 3 the error is same as node 2 on rotation, this errors on displacement's section, on force's section there is error valued with 0.72% in node 1 and node 3, if we gather all the errors we have a total error valued with 1.825%, the total error is under 2% so we notice that the proposed program is reliable to solve a simple beam structures.

### 3.6 Comparing program results with RDM6 results in complex cases

When evaluating the performance of a developed program for structural analysis, it's often necessary to compare its results with those obtained from simulations using established software such as RDM6. This comparison is particularly important in the case of complex structures where manual calculations are difficult or even impossible. With this in mind, this part of the work examines the comparison between the results produced by the developed program and those from simulations conducted with RDM6, highlighting the discrepancies and similarities between these two analysis approaches for more complex structural cases.

#### 3.6.1 Illustration of the computation for a truss element

Given a truss system consisting of 7 bars with a double support at node 5 and a single support at node 2, with identical lengths  $L=1$  m meter and the same stiffness  $E=2 \times 10^5$  MPa, and cross-sections  $A=10$  cm<sup>2</sup>, and subjected to concentrated forces as shown in the figure:  $F_1=20$ KN and  $F_2=10$ KN. We propose to determine the unknown displacements and the support reactions.



**Figure 3.23:** A truss structure consisting of 7 bars and 5 nodes

Using the calculation program dedicated to 2D bar elements (trusses), and inputting the data from this problem, we arrived at the results shown in the [Table 3.12](#).

**Table 3.12:** Results provided by the developed calculation program for the truss structure consisting of 7 bars

	coordinate_x	coordinate_y	displacement_x	displacement_y	load_x	load_y
0	0	0	0.0000	0.0000	0	17500
1	1000	0	0.0875	-0.2561	0	0
2	2000	0	0.1500	0.0000	0	12500
3	1500	500	0.0429	-0.1944	0	-10000
4	500	500	0.1186	-0.2423	0	-20000

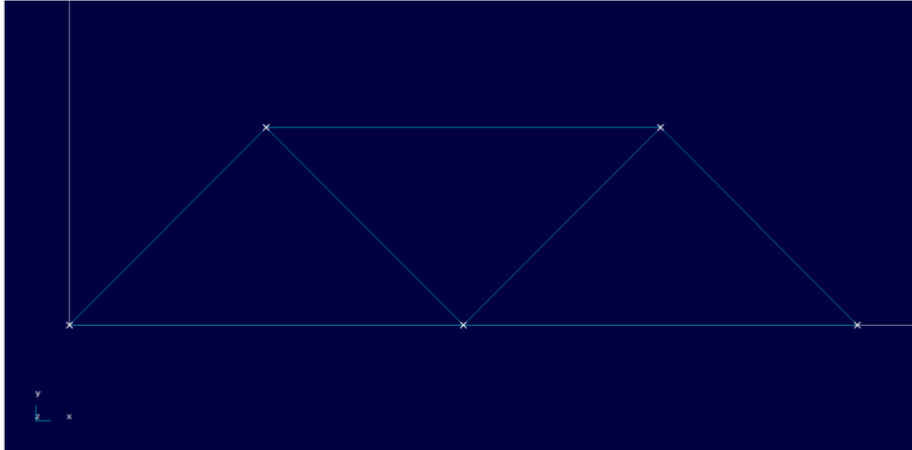
To compare the results obtained with others provided by the RDM6 software for this structure of 7 bars, we present the different steps of the simulation:

### 1- Initial Configuration

- Open RDM6.
- Configure the basic settings based on the measurement units you use (mm).

### 2- Creating the truss model

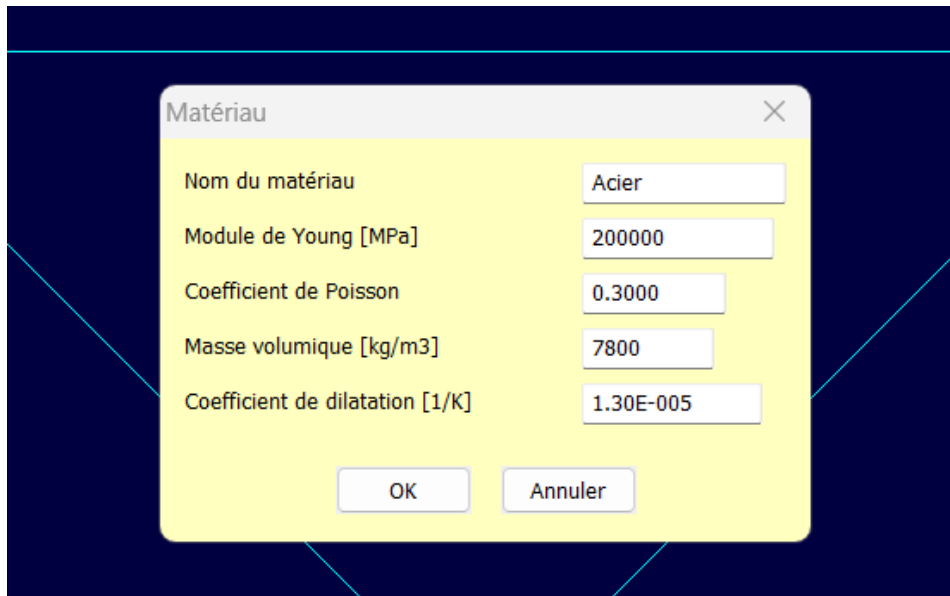
- Node definition: access the model creation section in RDM6 and add the structure's nodes by specifying their coordinates (X, Y).
- Adding bars: Select the start and end nodes for each bar and add the 7 truss bars by connecting the appropriate nodes ([Figure 3.24](#)).



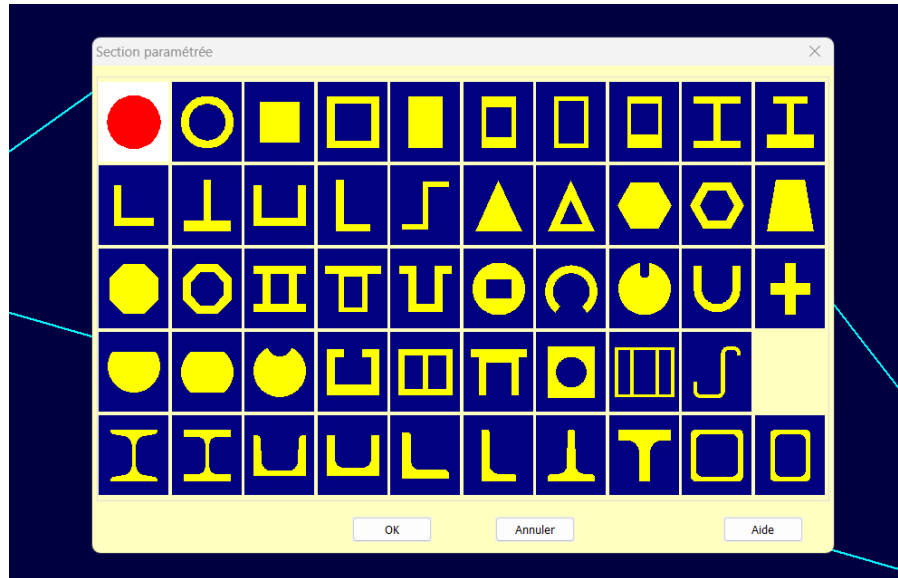
**Figure 3.24:** Creating the truss model consisting of 7 bars and 5 nodes

### 3- Defining material and section properties

- Material properties definition: Enter the properties of the material used (Young's modulus, Poisson's ratio, etc.) (Figure 3.25).
- Defining bar sections: Specify the dimensions of the bar sections (Figure 3.26) (cross-sectional area, moment of inertia, etc.).



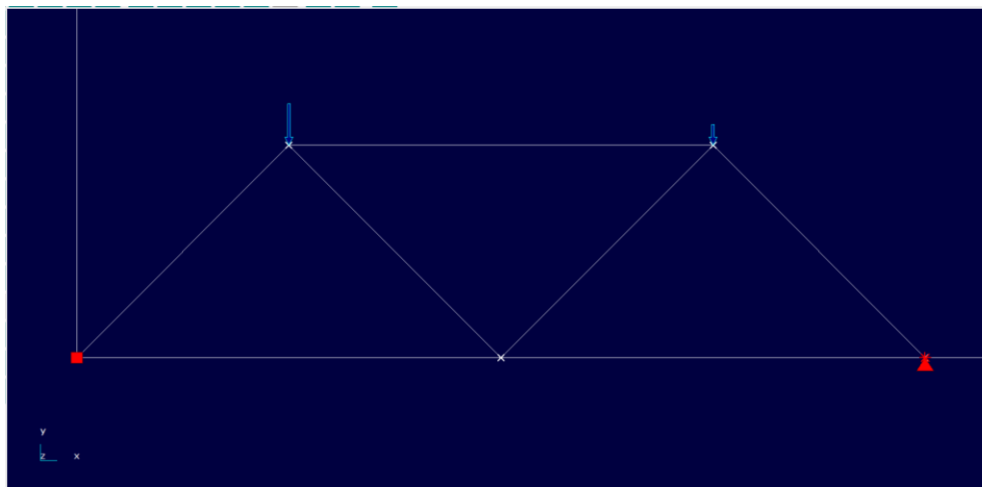
**Figure 3.25:** Properties of the material used



**Figure 3.26:** Dimensions and shape of the chosen section of the bar

#### 4- Applying loads and boundary conditions

- Applying Loads: Add the forces applied to the nodes or bars and specify the direction and intensity of the loads (Figure 3.27).
- Defining Boundary Conditions: Fix the support nodes by defining their restricted degrees of freedom (DOF) and indicate nodes that are fixed or have limited displacements (Figure 3.27).



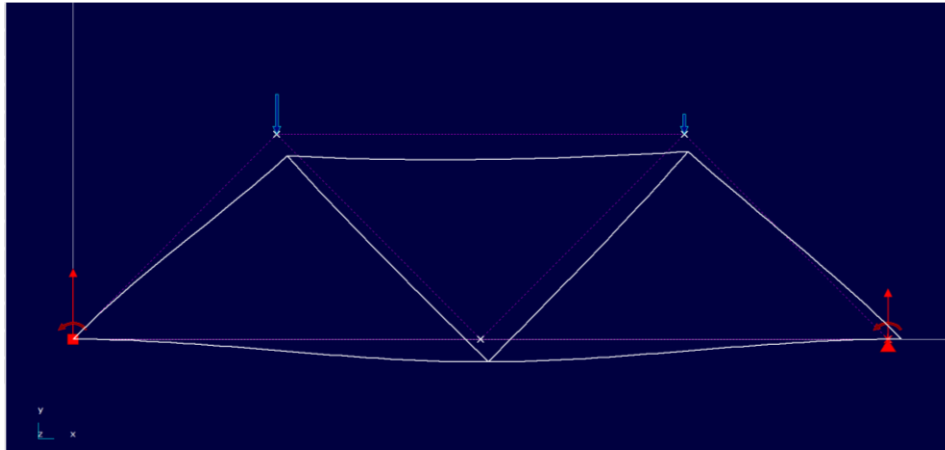
**Figure 3.27:** Applying loads and boundary conditions

#### 5- Performing the Analysis

- Running the Simulation: Verify that all data is correctly entered and execute the analysis to obtain results (reactions, displacements, internal forces).

## 6- Analyzing the Results

- Viewing the Results: Review the results provided by RDM6, including node displacements and internal forces in the bars and compare these results with those obtained by our developed calculation program.
- Reports and Export: Generate detailed reports from the results and export data if necessary for further analysis.



**Figure 3.28:** Viewing the Results

```

File Edit Format View Help
+-----+
| Results |
+-----+

+-----+
| Nodal displacements [ mm ] |
+-----+

Node 1    Ux = 0.000E+00          Uy = 0.000E+00
Node 2    Ux = 8.741E-02          Uy = -2.558E-01
Node 3    Ux = 1.498E-01          Uy = 0.000E+00
Node 4    Ux = 4.371E-02          Uy = -1.944E-01
Node 5    Ux = 1.186E-01          Uy = -2.423E-01

+-----+
| Support reaction(s) [ N ] |
+-----+

Node 1    Rx = 0.0                Ry = 17500.6
Node 2    Fx = 0.0                Fy = 0.0
Node 3    Rx = 0.0                Ry = 12499.4
Node 4    Fx = 0.0                Fy = -10000.0
Node 5    Fx = 0.0                Fy = -20000.0

```

**Figure 3.29:** Exporting the report results

The results obtained from the simulations conducted using the RDM6 software for this truss structure composed of 7 bars are presented in [Table 3.13](#).

**Table 3.13:** Outcomes provided by RDM6 software for the truss structure consisting of 7 bars

Nodes	Displacement u (mm)	Displacement v (mm)	Forces/x-axis	Forces/y-axis (N)
Node 1	0	0	0	17500.6
Node 2	0.08741	-0.2558	0	0
Node 3	0.1498	0	0	12499.4
Node 4	0.04371	-0.1944	0	-10000
Node 5	0.1186	-0.2423	0	-20000

To compare these findings, the relative discrepancy calculated using formula (3.1) is shown in [Table 3.14](#).

**Table 3.14:** Relative error between the results generated by the RDM6 software and those produced by the proposed program for the truss structure consisting of 7 bars

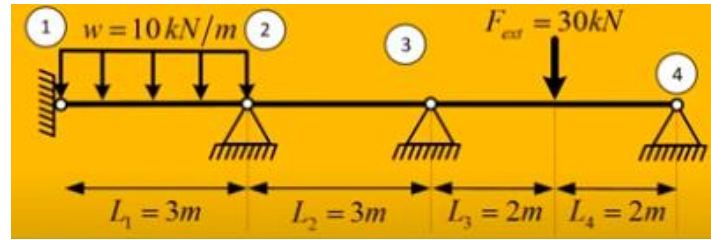
Nodes	(e%) u	(e%) v	(e%) F /x-axis	F /y-axis
Node 1	0	0	0	0.004
Node 2	0.1	0.05	0	0
Node 3	0.13	0	0	0.005
Node 4	1.9	0	0	0
Node 5	0	0	0	0

Table 3.14 presents the relative errors between RDM6's results and the proposed program's results for a truss consisting of 7 bars and 5 nodes with a square cross-section; we observe that in node 2 on displacement **u** there is an error that valued with 0.1%, on displacement **v** the error is 0.05%, in node 3 there is error on displacement **u** rated with 0.13%, node 4 has error as same as node 3 with a value of 1.9%, this errors on displacement's section, on force's section there is error valued with 0.004% in node 1 and 0.005% in node 3, if we gather all the errors we have a total error valued with 2.19%, the total error is under 5% so we can say that the proposed program is usable comparing with RDM6 software in complex truss structures.

### 3.6.2 Illustration of the computation for a beam element

Given a beam with a solid rectangular cross-section (width  $b=10\text{cm}$  and height  $h=20\text{ cm}$ ), fixed at node 1, and with a double support at nodes 2, 3, and 4, with lengths  $L_1=L_2=3\text{ meters}$ ,  $L_3=L_4=2\text{ meters}$ . The beam is subjected to a bending load  $F=30\text{ KN}$  and a uniformly distributed linear load ( $w=10\text{ KN/m}$ ) applied on element 1 [Figure 3.30](#). We propose to determine the unknown bend, rotations, reaction, and Moments. Given:  $E=2\times 10^5\text{Mpa}$ .





**Figure 3.30:** A beam structure consisting 5 nodes

Using the calculation program dedicated to 2D beam elements, and inputting the data from this problem, we arrived at the results shown in the [Table 3.15](#).

**Table 3.15:** Results provided by the developed calculation program for the beam structure

coordinate_x	coordinate_y	displacement_y	rotation	load_y	torque
0	0	0	0.000000	0.0	15000000
1	3000	0	0.000000	0.0	-7500000
2	6000	0	0.000000	0.0	15000000
3	8000	0	-0.749825	0.0	-30000
4	10000	0	0.000000	0.0	-15000000

To compare the results obtained with others provided by the RDM6 software for this structure of 7 bars, we present the different steps of the simulation:

### 1- Initial Configuration

- Open RDM6.
- Configure the basic settings based on the measurement units you use (mm).

### 2- Creating the truss model

- Node definition: access the model creation section in RDM6 and add the structure's nodes by number of nodes and specifying their coordinates in axis X.
- Adding beams: the beams are defined automatically after defining the nodes ([Figure 3.31](#)).



**Figure 3.31:** Creating the beam model consisting 5 nodes

### 3- Defining material and section properties

- Material properties definition: Enter the properties of the material used (Young's modulus, Poisson's ratio, etc.) (Figure 3.32).
- Defining beam sections: Specify the dimensions of the beam sections (Figure 3.33)

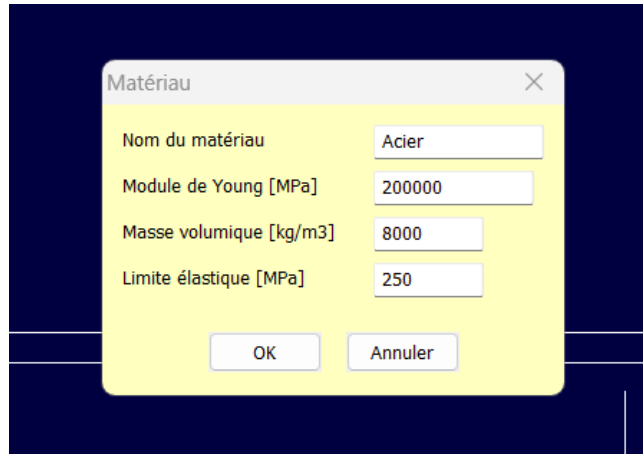


Figure 3.32: Properties of the material used

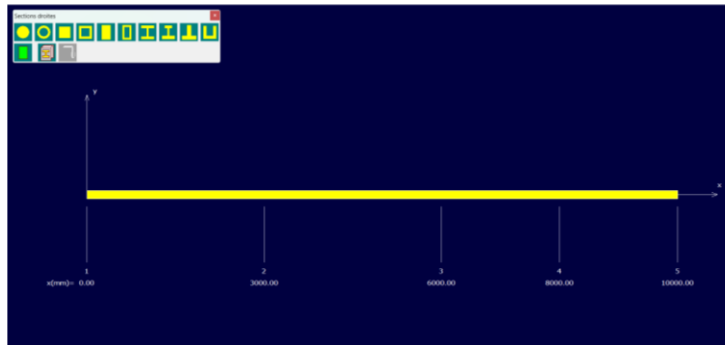
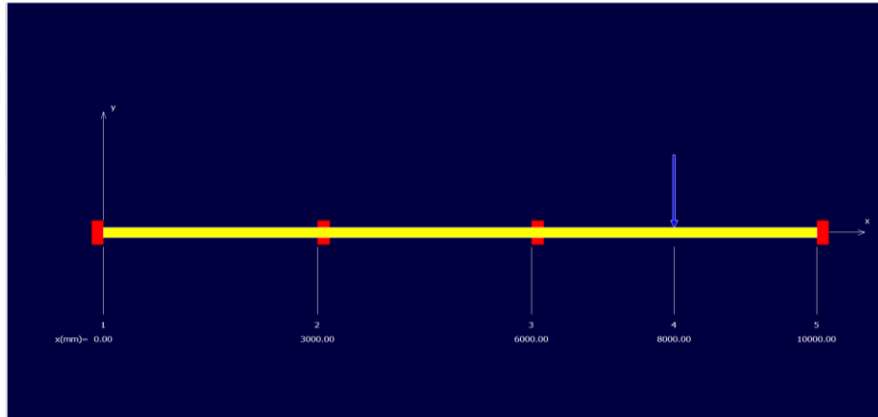


Figure 3.33: Dimensions and shape of the chosen section of the beam

### 4- Applying loads and boundary conditions

- Applying Loads: Add the forces applied to the nodes and specify the direction and intensity of the loads (Figure 3.34).
- Defining Boundary Conditions: Fix the support nodes by defining their restricted degrees of freedom (DOF) and indicate nodes that are fixed or have limited displacements (Figure 3.34).



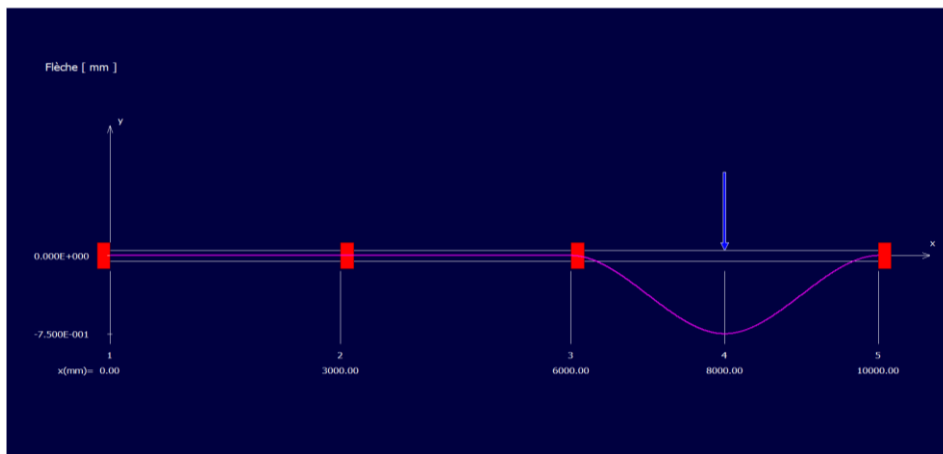
**Figure 3.34:** Applying loads and boundary conditions

### 5- Performing the Analysis

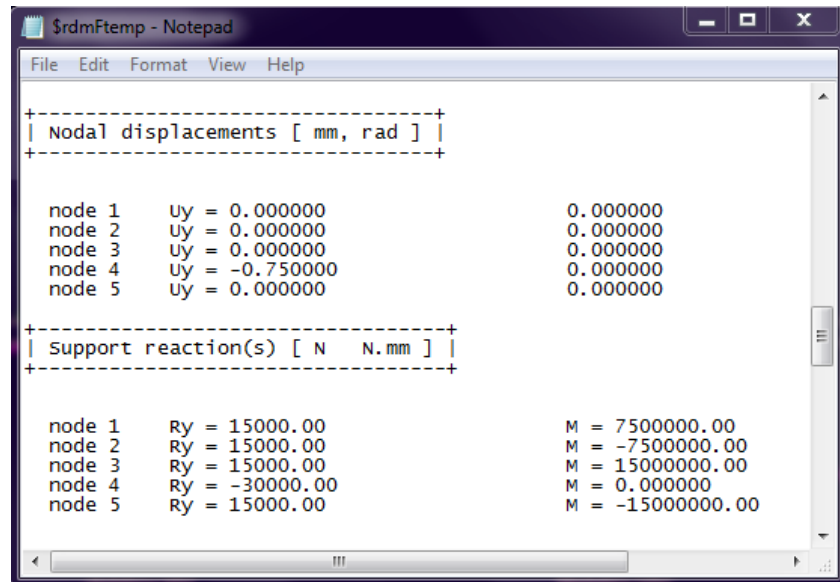
- **Running the Simulation:** Verify that all data is correctly entered and execute the analysis to obtain results (bend, rotations, reaction, and Moments).

### 6- Analyzing the Results

- **Viewing the Results:** Review the results provided by RDM6, including displacements and rotation and reaction and torque in the nodes and compare these results with those obtained by our developed calculation program.
- **Reports and Export:** Generate detailed reports from the results and export data if necessary for further analysis.



**Figure 3.35:** Viewing the Results



```

+-----+
| Nodal displacements [ mm, rad ] |
+-----+

node 1   uy = 0.000000           0.000000
node 2   uy = 0.000000           0.000000
node 3   uy = 0.000000           0.000000
node 4   uy = -0.750000          0.000000
node 5   uy = 0.000000           0.000000

+-----+
| Support reaction(s) [ N  N.mm ] |
+-----+

node 1   Ry = 15000.00           M = 7500000.00
node 2   Ry = 15000.00           M = -7500000.00
node 3   Ry = 15000.00           M = 15000000.00
node 4   Ry = -30000.00          M = 0.000000
node 5   Ry = 15000.00           M = -15000000.00

```

**Figure 3.36:** Exporting the report results

The results obtained from the simulations conducted using the RDM6 software for this beam structure is presented in [Table 3.16](#).

**Table 3.16:** Outcomes provided by RDM6 software for the beam structure

Nodes	Displacements		Loads	
	bend (mm)	rotation (rad)	Forces (N)	Moments (N.mm)
Node 1	0	0	15000	7500000
Node 2	0	0	15000	-7500000
Node 3	0	0	15000	15000000
Node 4	-0.75	0	-30000	0
Node 5	0	0	15000	-15000000

To compare these findings, the relative discrepancy calculated using formula (3.1) is shown in [Table 3.17](#).

**Table 3.17:** Relative error between the results generated by the RDM6 software and those produced by the proposed program for the beam structure

Nodes	(e%) displacements		(e%)loads	
	bend	rotation	Forces	Moments
Node 1	0	0	0	0
Node 2	0	0	0	0
Node 3	0	0	0	0
Node 4	0.023	0	0	0
Node 5	0	0	0	0

Table 3.17 presents the relative errors between RDM6's results and the proposed program's results for a beam contains 4 elements and 5 nodes with a rectangular cross-section; we observe

that in node 4 on bend there is an error that valuated with 0.023%, and in the other nodes there is no error and this show us that the proposed program almost precise as RDM6- software in complex beams structures.

### **3.7 Conclusion**

In this chapter, we achieve our goal which is programming a program that helping us with calculating in finite element method and it making complex structures solved easily, we began with bar program, next we programming a program for truss, then we beam's program, and we validate our programs with simple examples that solved manually, we used RDM6 software for validating our program for the complex examples. Finally, we used error formula to see the difference between the proposed program's results and validation's results. After using error formula we assured that the proposed programs are usable and helpful with the finite element method, and proposed program's results is reliable.

## *General Conclusion*

In conclusion, this dissertation has explored the increasing importance of Mechanical construction in various aspects of our lives. However, the process of building mechanical systems is often challenging, particularly when it comes to the presence of complex systems. To address this issue, an efficient program has been developed as part of this research. The developed program has demonstrated its capability to accurately calculate and export Finite Element Method results. This process significantly simplifies the Finite Element Method calculations. The results obtained from the developed program show its potential for practical application in various fields, such as construction, architecture, civil engineering, and virtual simulations. The accurate results of the program enable engineers, students, and decision-makers to focus on the core aspects of mechanical construction without the hindrance of long inaccurate results. Our research not only presents a novel approach to mechanical construction but also emphasize the importance of automation and efficiency in the study of mechanical systems. Because there is no such completed or perfect research, ours still holds a lot of potential for the future. Adding a proper user interface and merging the 3 systems into one major program instead of 3 separate ones, we might also think about adding a graphical input of elements and nodes, where the user can draw the system directly with a set of tools and features.

## References

- [1] Lu, C., Gao, Q., Fu, C., & Yang, H. (2017). Finite Element Method of BBM-Burgers Equation with Dissipative Term Based on Adaptive Moving Mesh, *Discrete Dynamics in Nature and Society*, 2017(1), 3427376
- [2] Chandrupatla, T. R., Belegundu, A. D., Ramesh, T., & Ray, C. (2012). *Introduction to finite elements in engineering*, London: Pearson.
- [3] Samir, D. E. G. H. B. O. U. D. J. METHODE DES ELEMENTS FINIS.
- [4] ZEROUROU. Bachir., OUDJEDI. Lydia. ; (2016) Programmation de la méthode des éléments finis sous MATAB
- [5] RDM6
- [6] Kattan, P. I. (2010). *MATLAB guide to finite elements: an interactive approach*. Springer Science & Business Media.
- [7] Saleh, Faisal & Benterboua, Farah. (2016). Elaboration d'un code éléments finis sous Matlab pour l'analyse statique des structures en treillis plans articulés.
- [8] Wahid. KADDOURI ; METHODE DES ELEMENTS FINIS ANALYSE DES STRUCTURES PAR ELEMENTS FINIS ; Batna, 2017
- [9] SEGHIR, A. (2005). Cours, << Méthode des Éléments Finis>>. Université Abderrahmane Mira–Bejaia, Faculté de Technologie, Département de Génie Civil, 2014.
- [10] Belkheira, R., & Belhouari, F. (2021). Modélisation du comportement statique d'une poutre FGM par la méthode des éléments finis (Doctoral dissertation, FACULTÉ DES SCIENCES APPLIQUÉES DÉPARTEMENT GÉNIE CIVIL).
- [11] Gries, D. (2012). *The science of programming*, Springer Science & Business Media
- [12] Charpentier, M. (2022). *Functional and Concurrent Programming: Core Concepts and Features*. Addison-Wesley Professional.

[13] Gorodniaia, L., & Andreyeva, T. (2016). Study of Programming Paradigms, In INTED2016 Proceedings (pp, 7482-7491), IATED

[14] Van Horn II, B. M., & Nguyen Q. (2023), Hands-On Application Development with PyCharm: Build applications like a pro with the ultimate python development tool. Packt Publishing Ltd.

[15] Hameed, A. (2016). Software development lifecycle for extreme programming; international Journal of Information Technology and Electrical Engineering, 5(1), 7-13

[16] Fourment, M., & Gillings, M. R. (2008). A comparison of common programming languages used in bioinformatics. BMC bioinformatics, 9, 1-9

[17] SALEEM, M., & ASHRAF, M. CHALLENGES AND TRENDS OF EMERGING PROGRAMMING LANGUAGES: A SYSTEMATIC.

[18] Python, W. (2021). Python, Python releases for windows, 24.

[19] Mukherjee, S., Almanza, A., & Rubio-González, C. (2021, July). Fixing dependency errors for Python build reproducibility. In Proceedings of the 30th ACM SIGSOFT international symposium on software testing and analysis (pp, 439-451)

[20] Randles, B. M., Pasquetto, I. V., Golshan, M. S., & Borgman, C. L. (2017, June). Using the Jupyter notebook as a tool for open science: An empirical study, In 2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL) (pp, 1-2), IEEE

[21] Roger, D., Frédéric, L., Marc, R. (2005, August), Découvrir le calcul de structure avec le Progiciel RDM 6.