# Element Stiffness Matrix Derivation of Rod Element

**Dogukan Gedik**

January 27, 2025

# Contents

# List of Figures

# Introduction

The objective of this report is to derive the element stiffness matrix of a bar/rod element using MATLAB. Unlike conventional approaches, where the modulus of elasticity and the cross-sectional area are assumed constant, this study considers these parameters as functions varying along the longitudinal axis of the element. Consequently, the governing equation becomes more complex, resulting in terms that are difficult to integrate analytically. To address this, Gaussian-Legendre numerical integration has been implemented in MATLAB, enabling an approximate solution for the integral.

# Chapter 1

# Theoretical Background

## 1.1 What is Rod/Bar Element

Rod/bar elements are one-dimensional finite elements used to model axial deformation in structural members. The governing equation for such elements derives from the equilibrium of forces along their length. The primary equation is [5]:

$$\frac{d}{dx}\left(AE\frac{du}{dx}\right) + f(x) = 0, \tag{1.1}$$

where:

- $A$: Cross-sectional area,
- $E$: Young's modulus of the material,
- $u$: Displacement along the axis,
- $f(x)$: Axial distributed load.

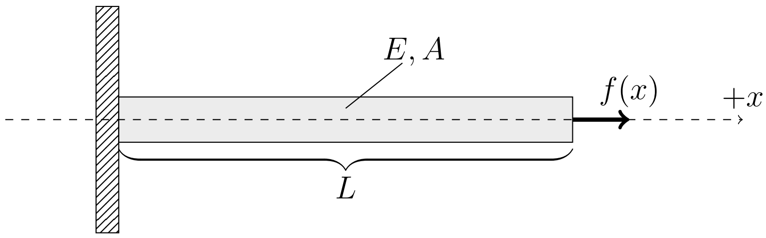



**Figure 1.1:** Representation of a bar element with varying properties.

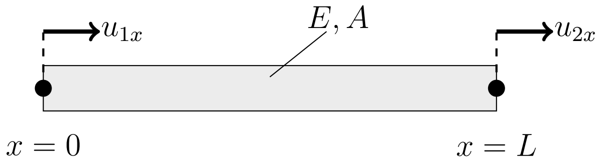



**Figure 1.2:** Bar element with 2 node representation.

$$K^e = \frac{EA}{L} \begin{array}{cc} & \begin{array}{cc} u_{1x} & u_{2x} \end{array} \\ & \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{array}{c} u_{1x} \\ u_{2x} \end{array} \end{array} \tag{1.2}$$

Equation 1.2 is a basic element stiffness matrix of a bar element in constant E and A condition.

## 1.2 What is Shape Function

Shape functions interpolate the solution between the nodes of an element. They are defined based on the type and number of nodes.

### 1.2.1 Linear Element Shape Functions

The shape functions for a 2-node element are as shown below. The shape functions are illustrated on the element in Figure1.3 [5].

$$N_1(\xi) = \frac{1-\xi}{2},$$
$$N_2(\xi) = \frac{1+\xi}{2},$$

where $\xi$ is the local coordinate.

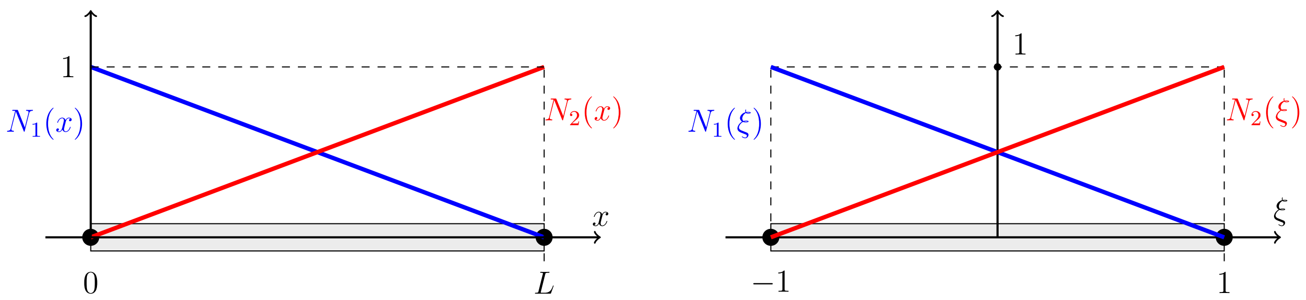



**Figure 1.3:** Linear shape functions: Replacing the global coordinate x by the local one $\xi$.

### 1.2.2 Quadratic Element Shape Funtions

The shape functions for a 3-node element are as shown below. The shape functions are illustrated on the element in Figure1.4 [5].

$$N_1(\xi) = \frac{1}{2}\xi(\xi-1),$$
$$N_2(\xi) = 1-\xi^2,$$
$$N_3(\xi) = \frac{1}{2}\xi(\xi+1).$$

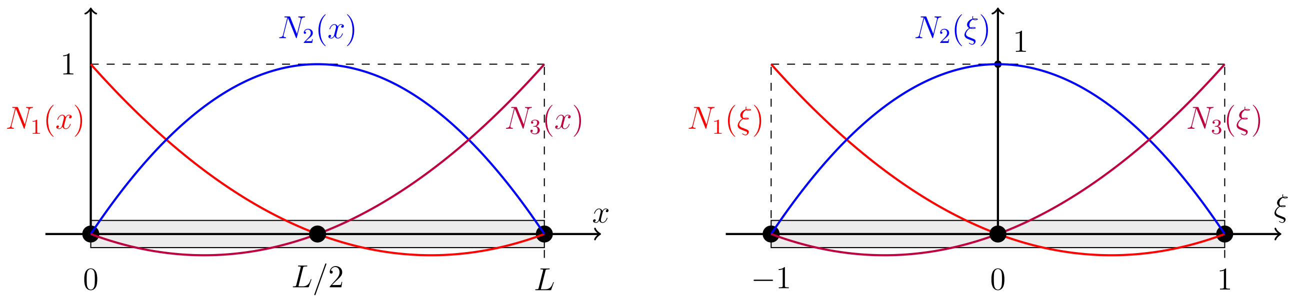



**Figure 1.4:** Quadratic shape functions: Replacing the global coordinate x by the local one $\xi$.

### 1.2.3 Cubic Element Shape Functions

The shape functions for a 4-node element are as shown below. The shape functions are illustrated on the element in Figure1.5 [3] [4]:

$$N_1(\xi) = \frac{9}{16}(1-\xi)(\xi^2 - \frac{1}{9}),$$
$$N_2(\xi) = \frac{27}{16}(\xi^2 - 1)(\xi - \frac{1}{3}),$$
$$N_3(\xi) = \frac{27}{16}(1-\xi^2)(\xi + \frac{1}{3}),$$
$$N_4(\xi) = \frac{9}{16}(\xi+1)(\xi^2 - \frac{1}{9}).$$

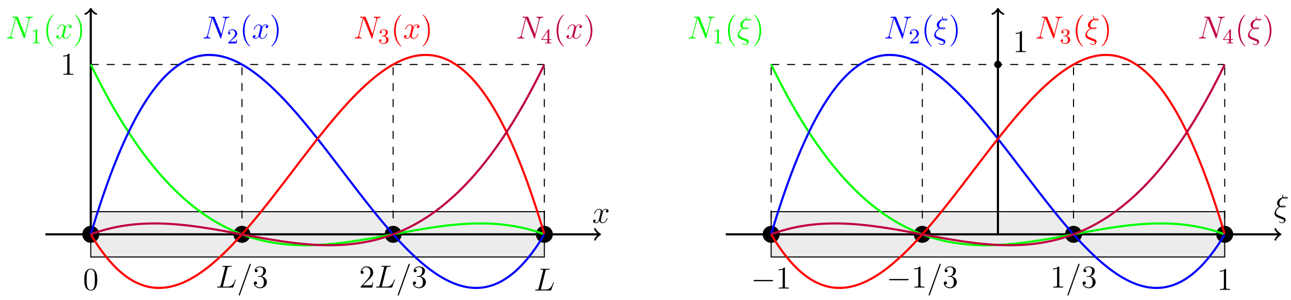



**Figure 1.5:** Cubic shape functions: Replacing the global coordinate x by the local one $\xi$.

## 1.3 Governing Equation and Stiffness Matrix

The governing equation leads to the element stiffness matrix. For a bar element with $A(x)$ and $E(x)$ varying along the length, the general stiffness matrix formula is: [4]

$$K_{ij}^e = \int_{-1}^{1} \frac{1}{J} \frac{dN_i}{d\xi} EA \frac{dN_j}{d\xi} d\xi \tag{1.3}$$

where $N_i$ and $N_j$ are shape functions, and $J$ is the Jacobian determinant.

## 1.4 Numerical Integration

To solve the integral numerically, Gaussian quadrature is used. It approximates the integral as a weighted sum of function evaluations at specific integration points:

$$\int_{-1}^{1} f(\xi)d\xi \approx \sum_{i=1}^{n} w_i f(\xi_i) \tag{1.4}$$

where $w_i$ are weights and $\xi_i$ are integration points. Common integration points and weights for different orders are shown in the table below:

**Table 1.1:** Gaussian Quadrature Points and Weights

| Order | Integration Points $\xi_i$ | Weights $w_i$ |
|---|---|---|
| 1 | 0 | 2.0 |
| 2 | $\pm 0.577$ | 1.0 |
| 3 | $0, \pm 0.774$ | $0.889, 0.555$ |
| 4 | $\pm 0.339, \pm 0.861$ | $0.652, 0.348$ |

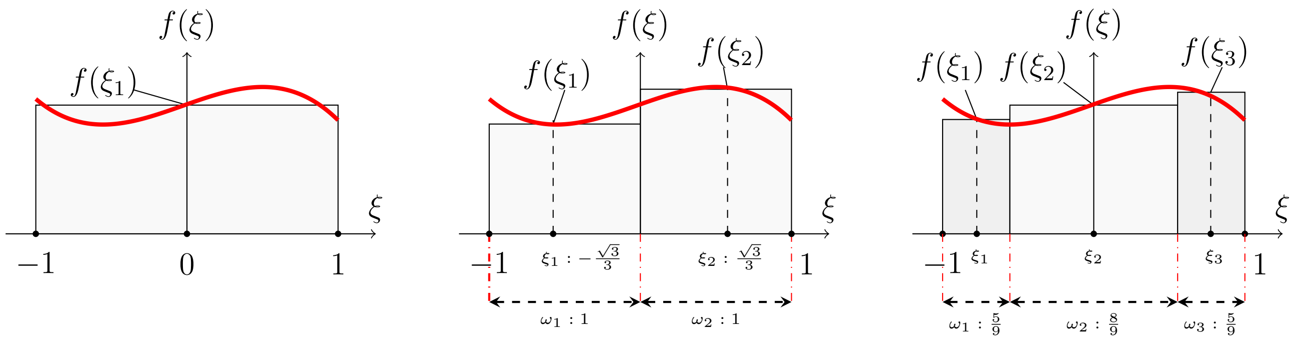



**Figure 1.6:** Gaussian quadrature:Integration points and weights representation

### 1.4.1 Demonstration of the Gauss–Legendre Procedure

Lets calculate below integral:

$$I = \int_{2}^{4} (x^2 + 5x - 2)dx \tag{1.5}$$

This integral can be solved analytically and to the solution is $I = 44.6666667$. If we want to calculate numerically (Gauss-Legendre), first we need to change integration boundaries to [-1,1]. [?]

$$x = \frac{(a+b)}{2} + \frac{(a-b)}{2}\lambda = \frac{(4+2)}{2} + \frac{(4-2)}{2}\lambda = 3 + \lambda$$

$$dx = \frac{(a-b)}{2}d\lambda = \frac{(4-2)}{2}d\lambda = d\lambda$$

$$I = \int_{2}^{4} (x^2 + 5x - 2)dx = \int_{-1}^{1} (2)[(3 + \lambda)^2 + 5(3 + \lambda) - 2]d\lambda \tag{1.6}$$

Now we can use Gauss-Legendre two point integration;

$$\omega_1 = \omega_2 = 1 \text{ and } \lambda_1 = 0.577, \lambda_2 = -0.577$$

$$I \approx \sum_{i=1}^{2} w_i f(\xi_i) = \omega_1 f(\lambda_1) + \omega_2 f(\lambda_2) \tag{1.7}$$

$$\begin{aligned} I &= (1)\left[(3 + (0.5777))^2 + 5(3 + (0.5777)) - 2\right] \\ &+ (1)\left[(3 + (-0.5777))^2 + 5(3 + (-0.5777)) - 2\right] \\ &= 44.6658580 \end{aligned}$$

The error in the obtained value arises from the rounding of the integration point value. This value is theoretically $\pm 1/\sqrt{3}$, with a numerical representation of $\pm 0.577350269$. When this exact value is used, the result of the numerical integration is obtained as 44.6666667.

# Chapter 2

# Programming Methodology

For solving the problem in this project, Matlab was chosen as the programming language. Matlab was selected due to its advantages in matrix operations and its user-friendly interface. The function-based programming paradigm was employed. This allows the element stiffness matrix to be calculated by calling appropriate functions corresponding to the number of nodes in the element within the code.

In the code, the modulus of elasticity ($E$) and cross-sectional area ($A$) values are defined as linear functions dependent on x. Thus, the coefficients a and b can be set to zero, allowing the element stiffness matrix to be computed with constant $E$ and $A$ values. This enables both constant and variable modeling, depending on the parameter settings.

The algorithm of the code was designed in accordance with the theoretical background presented in the report. After determining the appropriate shape functions based on the number of nodes, their derivatives with respect to $x$ were calculated, and integration was performed by including the Jacobian. The purpose of this step is to map from the global coordinate x to the local coordinate $\xi$.

The resulting integral was computed using Two-point Gauss-Legendre quadrature, and the element stiffness matrix was obtained as the final output.

## Algorithm Steps

1. **Input Problem Statements**
   All values representing the problem are provided as input. Additionally, the expressions E(x) and A(x) within the function are defined in a form suitable for the given problem.

2. **Set Gauss Points and Weights**
   In this step, the number of Gauss points and their corresponding coordinates are defined.

3. **Initialize Stiffness Matrix**
   The stiffness matrix $K$ is initialized as a zero matrix of appropriate dimensions.

4. **Loop through Gauss Points**
   For each Gauss point, the following computations are performed:

   (a) Compute shape function derivatives with respect to the natural coordinate. The derivatives of the shape functions with respect to the natural coordinate $\xi$ are evaluated.

(b) Compute the Jacobian. The Jacobian determinant, representing the transformation from the natural to the physical coordinate system, is computed.

(c) Compute the $B$ matrix. The $B$ matrix, which relates the strain to the displacement, is derived using the Jacobian.

(d) Compute the contribution to the element stiffness matrix. The element stiffness matrix is incremented by the weighted product of $B$ matrices and material properties.

5. **Display Result**
   Upon completing the loop, the final stiffness matrix $K$ is displayed.

6. **End**
   The algorithm concludes.

# Chapter 3

# Case Studies

## 3.1 Quadratic Bar Element with Constant E and A

For testing the code, a problem obtained from the source [5] has been used as a reference. The problem demonstrates the calculation of a 3-node quadratic element formulation. The values of $E$ and $A$ were assumed to be constant, and the integral below has been calculated.

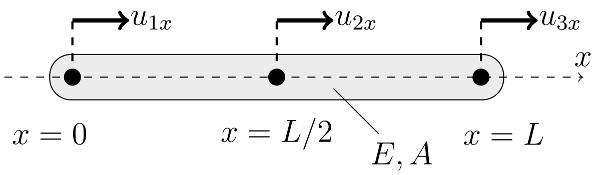



**Figure 3.1:** Quadratic bar element

$$K^e = EA \int_0^L \begin{bmatrix} \frac{dN_1(x)}{dx}\frac{dN_1(x)}{dx} & \frac{dN_1(x)}{dx}\frac{dN_2(x)}{dx} & \frac{dN_1(x)}{dx}\frac{dN_3(x)}{dx} \\ \frac{dN_2(x)}{dx}\frac{dN_1(x)}{dx} & \frac{dN_2(x)}{dx}\frac{dN_2(x)}{dx} & \frac{dN_2(x)}{dx}\frac{dN_3(x)}{dx} \\ \frac{dN_3(x)}{dx}\frac{dN_1(x)}{dx} & \frac{dN_3(x)}{dx}\frac{dN_2(x)}{dx} & \frac{dN_3(x)}{dx}\frac{dN_3(x)}{dx} \end{bmatrix} dx \tag{3.1}$$

In this case, $N_1(x)$, $N_2(x)$ and $N_3(x)$ are shape functions of quadratic bar element which already shown in Figure1.4 and derivatives of these functions;

$$\frac{dN_1}{dx} = -\frac{3}{L} + \frac{4x}{L^2}$$
$$\frac{dN_2}{dx} = \frac{4}{L} - \frac{8x}{L^2}$$
$$\frac{dN_3}{dx} = -\frac{1}{L} + \frac{4x}{L^2}$$

To verify the results through analytical computation, let us apply the integral to the element in the first row and first column of the matrix:

$$
\begin{aligned}
K^e_{1,1} &= EA\int_0^L \frac{dN_1(x)}{dx}\frac{dN_1(x)}{dx} \\
&= EA\int_0^L \left(-\frac{3}{L}+\frac{4x}{L^2}\right)\left(-\frac{3}{L}+\frac{4x}{L^2}\right) \\
&= EA\int_0^L \left(\frac{9}{L^2}-\frac{24x}{L^3}+\frac{16x^2}{L^4}\right) \\
&= EA\left(\frac{9x}{L^2}\Big|_0^L-\frac{24x^2}{2L^3}\Big|_0^L+\frac{16x^3}{3L^4}\Big|_0^L\right) \\
&= EA\left(\frac{9L}{L^2}-\frac{24L^2}{2L^3}+\frac{16L^3}{3L^4}\right) \\
&= EA\left(\frac{27}{3L}-\frac{36}{3L}+\frac{16}{3L}\right) \\
&= \frac{7EA}{3L}
\end{aligned}
$$

Equation 3.2 can be evaluated by analytical or numerical integration. After this calculation, the elemental stiffness matrix of the quadratic rod element as;

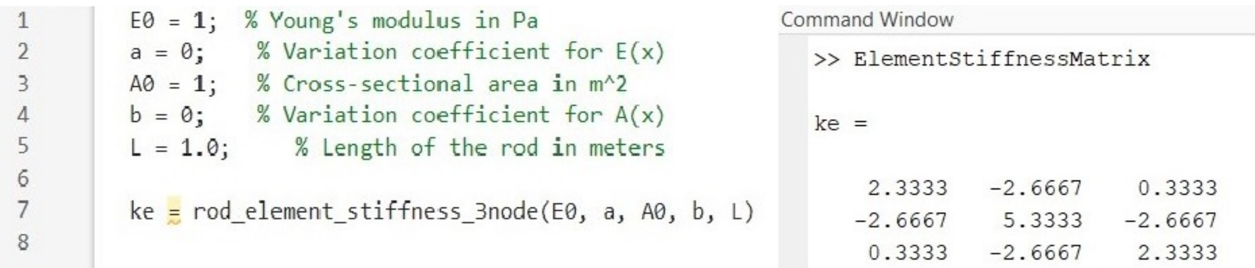
$$
K^e = \frac{EA}{3L}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix} dx \tag{3.2}
$$

To solve the problem using the developed code, the values of $E$, $A$, and $L$ should first be set to 1, as shown in Figure3.2. The coefficients $a$ and $b$, which define the functional dependence of $E$ and $A$, should be set to zero to represent constant values. Subsequently, the corresponding function should be called in the code as follows:

**Listing 3.1:** Funtion for 3-Node Element Stiffness Matrix

```
ke = rod_element_stiffness_3node(E0, a, A0, b, L)
```

After entering these inputs, running the program produces the result shown in Figure3.2.



```
E0 = 1;  % Young's modulus in Pa
a = 0;   % Variation coefficient for E(x)
A0 = 1;  % Cross-sectional area in m^2
b = 0;   % Variation coefficient for A(x)
L = 1.0;    % Length of the rod in meters

ke = rod_element_stiffness_3node(E0, a, A0, b, L)
```

```
Command Window
>> ElementStiffnessMatrix

ke =

    2.3333   -2.6667    0.3333
   -2.6667    5.3333   -2.6667
    0.3333   -2.6667    2.3333
```

**Figure 3.2:** Code implementation and output for quadratic element

Thus, the functionality of the implemented code has been verified. A similar approach can be applied for cases where the values of $E$ and $A$ are defined as linear functions of $x$.

## 3.2 Linear Rod Element with Functional E and A

Let's apply the code to the variable cross-sectional area (Linear A(x)) problem discussed in Prof. Öchsner's lecture. The problem representation is as follows:

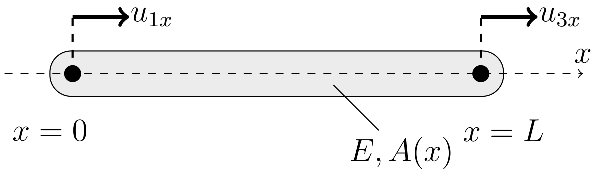



**Figure 3.3:** Linear rod element with changing cross-sectional area

To solve the problem, it is necessary to modify the mathematical expression representing the variation in the cross-sectional area within the linear bar element function. For this purpose, the expression has been defined as

$$A = A_0 \cdot (1 - bx).$$

The data input and function call are shown in Figure 3.4. Since the modulus of elasticity $E$ is assumed to be constant in this problem, the corresponding coefficient $a$ has been set to zero. The initial cross-sectional area $A_0$ is specified as 0.01 m$^2$, and the coefficient $b$ is set to 0.03.

In this case, $N_1(x)$ and $N_2(x)$ are shape functions of linear bar element which already shown in Figure1.3 and derivatives of these functions;

$$\frac{dN_1}{dx} = -\frac{1}{L}$$
$$\frac{dN_2}{dx} = \frac{1}{L}$$

According to lecture notes the element stiffness matrix of this problem states as:

$$K^e = \frac{E}{L} \cdot \frac{A_1 + A_2}{2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx \tag{3.3}$$

To verify the results through analytical computation, let us apply the integral to the element in the first row and first column of the matrix:

$$\begin{aligned} K^e_{1,1} &= E \int_0^L (A) \cdot \frac{dN_1(x)}{dx} \frac{dN_1(x)}{dx} \\ &= E \int_0^L [A_0 \cdot (1 + bx)] \cdot (-\frac{1}{L}) \cdot (-\frac{1}{L}) dx \\ &= E \int_0^L \left( \frac{A_0}{L^2} - \frac{A_0 bx}{L2} \right) \\ &= E \left[ \frac{A_0 x}{L^2} \bigg|_0^L - \frac{A_0 bx^2}{L2} \bigg|_0^L \right] \\ &= E \left[ \frac{A_0(2 - bL)}{2L} \right] \end{aligned}$$

This result is exactly the same as the expression on the lecture slayts:

$$\left(\frac{E}{L} \cdot \frac{A_1 + A_2}{2}\right)$$

The difference arises from the way the cross-sectional area function is defined. It is possible to demonstrate this using random values in the custom-written function. When analyzed parametrically, it can be observed that the same result is obtained, as shown below:

$$\begin{aligned} &A_1 + \frac{x}{L}(A_2 - A_1) \\ &x = 0 \implies A_1 \\ &x = L \implies A_2 \end{aligned}$$

*Lecture Notes*

$$\begin{aligned} &A_0 \cdot (1 - bx) \\ &x = 0 \implies A_0 \\ &x = L \implies A_0 \cdot (1 - bL) \end{aligned}$$

*Our Calculation*

$$\implies \frac{E}{2L} \cdot (A_1 + A_2)$$

*Lecture Notes*

$$\begin{aligned} &A_{x=0} + A_{x=L} \\ &A_0 + (A_0 \cdot (1 - bL)) \\ &\implies \frac{E}{2L} \cdot (A_0 \cdot (2 - bL)) \end{aligned}$$

*Our calculation*

As a final step, the result can be verified by comparing both calculations with the given condition:

$$\begin{aligned} K^e_{1,1} &= \frac{E}{2L} \cdot (A_1 + A_2) \\ &= \frac{200 \cdot 10^9}{2 \cdot 1} \cdot (0.01 + 9.7 \cdot 10^{-3}) \\ &= 1.97 \cdot 10^9 \end{aligned}$$

*According to lecture notes and analytic calculations*

$$\begin{aligned} K^e_{1,1} &= \frac{E}{2L} \cdot (A_0 \cdot (2 - bL)) \\ &= \frac{200 \cdot 10^9}{2 \cdot 1} \cdot (0.01 \cdot (2 - 0.03 \cdot 1)) \\ &= 1.97 \cdot 10^9 \end{aligned}$$

*Our calculation in analytical way*

```
E0 = 200e9;  % Young's modulus in Pa
a = 0;    % Variation coefficient for E(x)
A0 = 0.01;   % Cross-sectional area in m^2
b = 0.03;    % Variation coefficient for A(x)
L = 1.0;     % Length of the rod in meters

ke = rod_element_stiffness(E0, a, A0, b, L)
```

```
Command Window
>> elementstiffnessmatrix2

ke =

   1.0e+09 *

    1.9700   -1.9700
   -1.9700    1.9700
```

**Figure 3.4:** Code implementation and output for linear element

The results clearly demonstrate that the analytical calculations align with the element stiffness matrix numerically computed using the written code.

# Bibliography

[1] Klaus-Jürgen Bathe. *Finite element procedures*. K.J. Bathe, 2014.

[2] Tirupathi R. Chandrupatla and Ashok D. Belegundu. *Introduction to finite elements in engineering*. Prentice Hall, 2011.

[3] Saeed Moaveni. *Finite Element Analysis: Theory and Application with ANSYS*. Pearson Education Limited, 2015.

[4] Eugenio Oñate. *Structural Analysis with the Finite Element Method Linear Statics Volume 1. Basis and Solids*. Springer, 2009.

[5] Andreas Öchsner. *Computational Statics and Dynamics: An Introduction Based on the Finite Element Method*. Springer Nature Switzerland AG, 2023.

# Appendix: Code

**Listing 3.2:** MATLAB Code for 2-Node Element Stiffness Matrix

```
function K = rod_element_stiffness_2node(E0, a, A0, b, L)

%   E0: Constant modulus of elasticity
%   a:  Coefficient for linear variation of E(x)
%   A0: Constant cross-sectional area
%   b:  Coefficient for linear variation of A(x)
%   L:  Length of the rod element

% 2-noded bar element (linear shape functions)

% Gauss points and weights for 2-point quadrature
gaussPoints = [-1/sqrt(3), 1/sqrt(3)];
weights = [1, 1];

% Initialize stiffness matrix (2x2 for 2 nodal elements)
K = zeros(2, 2);

% Loop over Gauss points
for r = 1:length(gaussPoints)
xi = gaussPoints(r);
Wr = weights(r);

% Shape function derivatives with respect to xi
dNdxi = [-0.5, 0.5];

% Jacobian (J) and its inverse
J = L / 2;
dNdx = dNdxi / J;

% Coordinate x at the Gauss point
x = (1 + xi) * L / 2;

% Material properties at Gauss point
% This definitions can be change for each problem
E = E0 * (1 + a* x);
A = A0 * (1 - b * x);

% B matrix
B = dNdx;

% Compute stiffness matrix contribution at this Gauss point
K = K + (J * (B' * E * A * B) * Wr);
end
end
```

**Listing 3.3:** MATLAB Code for 3-Node Element Stiffness Matrix

```
function K = rod_element_stiffness_3node(E0, a, A0, b, L)

%   E0: Base modulus of elasticity
%   a: Coefficient of E(x)
%   A0: Base cross-sectional area
%   b: Coefficient of A(x)
%   L: Length of the rod element

% 3-noded bar element (quadratic shape functions)

% Gauss points and weights for 3-point quadrature
gaussPoints = [-sqrt(3/5), 0, sqrt(3/5)];
weights = [5/9, 8/9, 5/9];

% Initialize stiffness matrix (3x3 for 3 nodal elements)
K = zeros(3, 3);

% Loop over Gauss points
for r = 1:length(gaussPoints)
xi = gaussPoints(r);
Wr = weights(r);

% Shape function derivatives with respect to xi
dNdxi = [(xi - 0.5), -2*xi, (xi + 0.5)];

% Jacobian (J) and its inverse
J = L / 2;
dNdx = dNdxi / J;

% Coordinate x at the Gauss point
x = (1 + xi) * L / 2;

% Material properties at Gauss point
% This definitions can be change for each problem
E = E0 *(1 + a*x);
A = A0 *(1 + b*x);

% B matrix
B = dNdx;

% Compute stiffness matrix contribution at this Gauss point
K = K + (J * (B' * E * A * B) * Wr);
end
end
```

**Listing 3.4:** MATLAB Code for 2-Node Element Stiffness Matrix Inputs

```
clear;clc;

E0 = 200e9;  % Young's modulus in Pa
a = 0;     % Variation coefficient for E(x)
A0 = 0.01;   % Cross-sectional area in m^2
b = 0.03;    % Variation coefficient for A(x)
L = 1.0;     % Length of the rod in meters

% Dont forget to change mathematical definition of E(X) and A(x)
% in the funtion definition.

K = rod_element_stiffness_2node(E0, a, A0, b, L)
```

**Listing 3.5:** MATLAB Code for 3-Node Element Stiffness Matrix Inputs

```
clear;clc;

E0 = 1;  % Young's modulus in Pa
a = 0;     % Variation coefficient for E(x)
A0 = 1;    % Cross-sectional area in m^2
b = 0;     % Variation coefficient for A(x)
L = 1.0;     % Length of the rod in meters

% Dont forget to change mathematical definition of E(X) and A(x)
% in the funtion definition.

K = rod_element_stiffness_3node(E0, a, A0, b, L)
```