ME621 - Advanced Finite Element Methods
Assignment 2

Tommaso Bocchietti

A.Y. 2023/24 - W24

UNIVERSITY OF
WATERLOO

Contents

1 Requests **3**

2 Methodology **3**

3 Solution **3**
3.1 Shape & Compatibility functions 4
3.2 Elemental matrices 6
3.3 Global matrices 7

4 Implementation **7**

5 Results **8**

A Mathematica code **9**

List of Figures

1 Problem representation 3
2 Element representation 3
3 Shape functions plotted from `Mathematica`. 5
4 Displacement of the right end of the bar vs. time 8

List of Tables

1 Parameters of the rod element. 3

1 Requests

A tensile force F is applied to the end of the rod as shown in Figure 1.

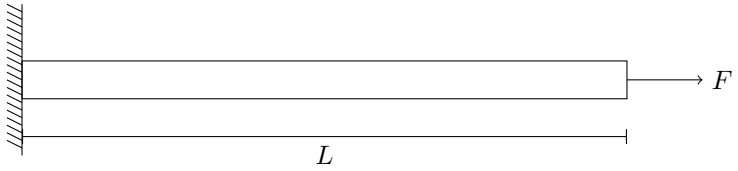

Figure 1: Problem representation

Discretize the rod into five 4-node elements to compute the displacement of the rod throughout time. Use an explicit integration scheme with Total Lagrangian finite element formulation. Use the parent (element) system of coordinates.
Use a linear constitutive equation of the form $P = E\frac{du}{dX}$, where P is the First Piola Kirchhoff stress.
The problem asks:

- Write out the shape functions for the following 4-node element with total length h_e and $\alpha = \beta = \frac{1}{3}$ in terms of the parent coordinate system, $-1 \leq \xi \leq 1$.
- Obtain the B_0^e matrix for a general 4-node element and write the elemental force vectors f_{int}^e, f_{ext}^e, f_{kin}^e, and elemental mass matrix M^e in discretized integral form. No need to integrate.
- Calculate the displacement of the node on the very right end after $t = 0.01s$ assuming the force is linearly increased at $100000kN/s$ from $0kN$ to $1000kN$. Plot the deformation of the right end of the bar vs time. The force is intentionally high and rapid in order to allow for a visual representation of the deformation behavior (ignore yielding and assume elasticity).
- Is there a difference in the deformation behavior if the $1000kN$ force is applied instantaneously at the beginning of the simulation (i.e. at $t = 0$)? Plot the deformation of the right end vs time.

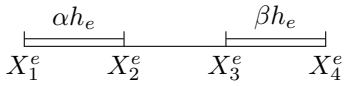

Figure 2: Element representation

Parameter	Value	Unit
E_0	70	GPa
ρ_0	2700	kg/m^3
A_0	300	mm^2
L_0	200	mm

Table 1: Parameters of the rod element.

2 Methodology

We will start by writing the analytical expression of the **shape functions** for the 4-node element with total length h_e and $\alpha = \beta = \frac{1}{3}$ in terms of the parent coordinate system, $-1 \leq \xi \leq 1$. To do so, we will make use of `Mathematica` to perform the symbolic computation and obtain the exact expression of the shape functions.
We will proceed by writing the elemental matrices (B_0^e, f_{int}^e, f_{ext}^e, f_{kin}^e) in terms of the parent coordinate system. We will then write the `MATLAB` code to solve the problem implementing the **explicit integration scheme with Total Lagrangian finite element formulation** and finally plot the deformation of the right end of the bar vs time, namely $u_{end}(t)$.

3 Solution

Here we present the all the calculations involved to answer the first two requests of the assignment.

3.1 Shape & Compatibility functions

Shape functions In general, the shape functions are defined as the interpolation polynomials that satisfy the Kronecker delta property:

$$N_i(X_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{1}$$

More over, another important property of the shape functions relative to the Finite Element Method is that their sum must be equal to 1 for any given element e:

$$\sum_{i=1}^{n} N_i^e(X) = 1 \tag{2}$$

We can write the shape functions in terms of both the local (parent) and global coordinates. As a general distinction, we have:

- **Local coordinates (parent)**: coordinates relative to the element, and are denoted by ξ.
- **Global coordinates**: coordinates relative to the entire domain, and are denoted by X.

We can now proceed to derive the shape functions for the 4-node element with total length h_e and $\alpha = \beta = \frac{1}{3}$ in terms of the parent coordinate system, $-1 \leq \xi \leq 1$.
Given the values of α and β, we can write the local coordinates values as:

$$X_1^e = 0 \tag{3}$$
$$X_2^e = \alpha h_e = \frac{1}{3} h_e \tag{4}$$
$$X_3^e = h_e - \beta h_e = \frac{2}{3} h_e \tag{5}$$
$$X_4^e = h_e \tag{6}$$

It's now useful to perform the mapping from the local X_i^e coordinates to the ξ coordinates. We know that the mapping is linear, and we can write the following system of equations:

$$X(\xi) = \frac{X_1 + X_2}{2} + \frac{X_2 - X_1}{2}\xi \tag{7}$$
$$\xi(X) = \frac{2}{X_2 - X_1}(X - \frac{X_1 + X_2}{2}) \tag{8}$$

By applying the above equations to the local coordinates X_i^e, we obtain the following values for the ξ coordinates:

$$\xi_1^e = -1 \tag{9}$$
$$\xi_2^e = -\frac{1}{3} \tag{10}$$
$$\xi_3^e = \frac{1}{3} \tag{11}$$
$$\xi_4^e = 1 \tag{12}$$

We can now use the Lagrange interpolation polynomials to obtain the shape functions for the given problem. The interpolations points are defined as:

$$P_{ij}^e = (\xi_j^e;\, y_{ij}^e) \quad j = 1, 2, 3, 4 \tag{13}$$
$$y_{ij}^e = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{14}$$

Where P_{ij}^e are the interpolation points for the i-th shape function of the e-th element, while j is the index of the interpolation point along the element.
For example, for the first shape function $N_1^e(\xi)$, we have the following interpolation points:

$$P_{11}^e = \xi_1^e, y_{11}^e = (-1;\, 1) \tag{15}$$

$$P_{12}^e = \xi_2^e, y_{12}^e = (-\frac{1}{3};\, 0) \tag{16}$$

$$P_{13}^e = \xi_3^e, y_{13}^e = (\frac{1}{3};\, 0) \tag{17}$$

$$P_{14}^e = \xi_4^e, y_{14}^e = (1;\, 0) \tag{18}$$

By interpolating the points above, we obtain the following shape functions:

$$N_1^e(\xi) = \frac{1}{16}(-9\xi^3 + 9\xi^2 + \xi - 1) \tag{19}$$

$$N_2^e(\xi) = \frac{9}{16}(\xi - 1)(\xi + 1)(3\xi - 1) \tag{20}$$

$$N_3^e(\xi) = -\frac{9}{16}(\xi - 1)(\xi + 1)(3\xi + 1) \tag{21}$$

$$N_4^e(\xi) = \frac{1}{16}(\xi + 1)(3\xi - 1)(3\xi + 1) \tag{22}$$

Those functions can also be visualized as in Figure 3.

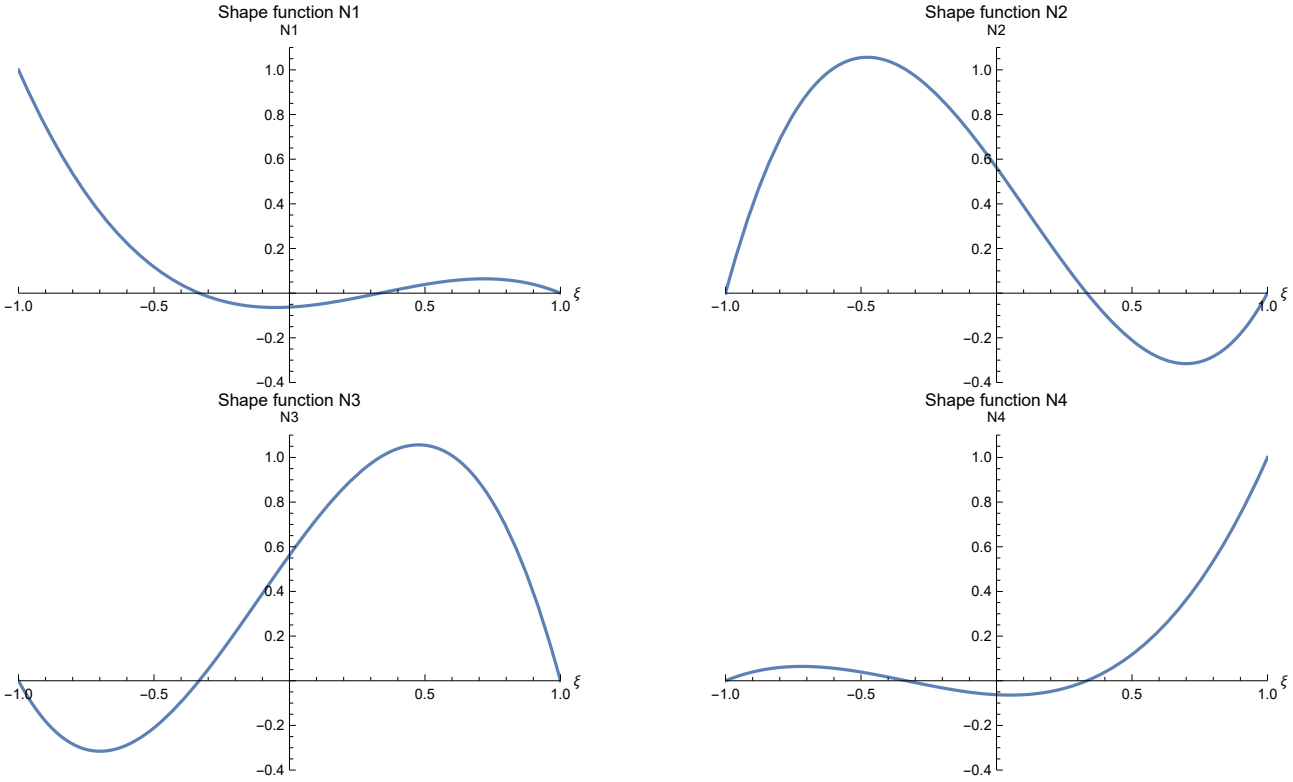

Figure 3: Shape functions plotted from `Mathematica`.

As we know shape functions can also be rearranged in the following matrix form:

$$\mathbf{N}^e(\xi) = \begin{bmatrix} N_1^e(\xi) & N_2^e(\xi) & N_3^e(\xi) & N_4^e(\xi) \end{bmatrix} \tag{23}$$

Compatibility functions During the derivation of the elemental matrices, we will also need to calculate the compatibility functions $B^e(\xi)$.
The compatibility functions are defined as the derivatives of the shape functions with respect to the global coordinates X. Since we have computed the shape functions in terms of the local coordinates ξ, we can use both the chain rule and the Jacobian matrix, or perform an inverse mapping from the local to the global coordinates and then differentiate.

$$B^e(\xi) = \frac{\partial N_i^e(\xi)}{\partial X} = \begin{cases} \frac{\partial N_i^e(\xi)}{\partial \xi}\frac{\partial \xi}{\partial X} & \text{using the chain rule} \\ \frac{\partial N_i^e(\xi=\xi(X)=\frac{2}{b-a}(X-\frac{a+b}{2}))}{\partial X} & \text{using the inverse mapping} \end{cases} \tag{24}$$

By adopting one of the two methods above, we can obtain the compatibility functions for the given problem as:

$$B_1^e(\xi) = \frac{9\xi(2-3\xi)+1}{8h_e} \tag{25}$$

$$B_2^e(\xi) = \frac{9(\xi(9\xi-2)-3)}{8h_e} \tag{26}$$

$$B_3^e(\xi) = -\frac{9(\xi(9\xi+2)-3)}{8h_e} \tag{27}$$

$$B_4^e(\xi) = \frac{9\xi(3\xi+2)-1}{8h_e} \tag{28}$$

In a matrix form, we can write the compatibility functions as:

$$\mathbf{B}^e(\xi) = \begin{bmatrix} B_1^e(\xi) & B_2^e(\xi) & B_3^e(\xi) & B_4^e(\xi) \end{bmatrix} \tag{29}$$

3.2 Elemental matrices

So far, we have derived the shape functions and the compatibility functions for the 4-node element in terms of the local coordinates system. We can now proceed to derive the elemental matrices for the given problem.
The elemental matrices (and vectors) are computed starting from previously derived shape functions and compatibility functions. Given that we have computed both the shape and compatibility functions in terms of the local coordinates system ξ, we will compute the elemental matrices in terms of the local coordinates system as well.
By definition, the elemental matrices are given by the following integrals:

$$f_{int}^e = \int_{X_1^e}^{X_n^e} B_0^{eT} P A_0 dX \tag{30}$$

$$f_{ext}^e = f_{body}^e + f_{traction}^e = \int_{X_1^e}^{X_n^e} N^{eT} \rho_0 b A_0 dX + [N^{eT} A_0 \tau\Big|_{X_1^e}^{X_n^e} \tag{31}$$

$$M^e = \int_{X_1^e}^{X_n^e} N^{eT} \rho_0 A_0 N^e dX \tag{32}$$

Where B_0^e and N^e are the matrices of the shape and compatibility functions in terms of the global coordinate system.
If we suppose to deal with a linear element (one that has a linear constitutive equation), then we can substitute the definition of $P = E\frac{du}{dX}$ (First Piola Kirchhoff stress, equivalent to engineering stress in 1D) in the definition of f_{int}^e and obtain:

$$f_{int}^e = \int_{X_1^e}^{X_n^e} B_0^{eT} E_0 A_0 \frac{du}{dX} dX = \tag{33}$$

$$= \int_{X_1^e}^{X_n^e} B_0^{eT} E_0 A_0 \frac{dN}{dX} u_{nodal} dX = \tag{34}$$

$$= \int_{X_1^e}^{X_n^e} B_0^{eT} E_0 A_0 B_0^e dX u_{nodal}^e \tag{35}$$

Where u_{nodal}^e is the vector of the nodal displacements of the element e.
Given that both B_0^e and N^e have been computed in terms of the local coordinates system ξ, then to perform the integrals we need to change the integration variable from X to ξ. To do so, we simply need to multiply the integrals by the Jacobian of the transformation, which is given by:

$$J = \frac{dX}{d\xi} = \frac{X_2 - X_1}{2} = \frac{h_e}{2} \tag{36}$$

Where h_e is the length of the element e.
Our final formulation for the elemental matrices is then given by:

$$f^e_{int} = \int_{-1}^{1} B_0^{eT} E_0 A_0 B_0^e J d\xi u^e_{nodal} = \int_{-1}^{1} B_0^{eT} E_0 A_0 B_0^e \frac{h_e}{2} d\xi u^e_{nodal} \tag{37}$$

$$f^e_{ext} = \int_{-1}^{1} N^{eT} \rho_0 b A_0 J d\xi + \left[N^{eT} A_0 \tau\right]_{-1}^{1} = \int_{-1}^{1} N^{eT} \rho_0 b A_0 J d\xi + \left[N^{eT} A_0 \tau\right]_{-1}^{1} \tag{38}$$

$$M^e = \int_{-1}^{1} N^{eT} \rho_0 A_0 N^e J d\xi = \int_{-1}^{1} N^{eT} \rho_0 A_0 N^e \frac{h_e}{2} d\xi \tag{39}$$

Where B_0^e, N^e and J are now expressed in terms of the local coordinates system ξ.

3.3 Global matrices

For the formulation of the FEM problem, we will need to compute the global matrices and vectors.
So far we have computed the elemental matrices in terms of the local coordinates system ξ. We can now proceed to assemble the global matrices and vectors.

Connectivity matrix The assembly of the global matrices and vectors is performed by looping over all the elements and adding the contribution of each element to the corresponding global matrix or vector. We can use the **connectivity matrix** to map the local degrees of freedom to the global degrees of freedom.
The connectivity matrix is also called Gather Scatter matrix and is defined as follows:

$$L_{ij} = \begin{cases} 1 & \text{if the i-th global degree of freedom is associated with the j-th local degree of freedom} \\ 0 & \text{otherwise} \end{cases} \tag{40}$$

We can also express the connectivity matrix for a given element using the Kronecker delta function as follows:

$$L^e_{ij} = \delta_{i,j} = \begin{cases} 1 & \text{if } j = (m-1)(e-1)+1 \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

Where m is the number of nodes per element and e is the element index. With a bit of algebra, we can state that the L matrix has a size of $m \times n$ where n is the number of global degrees of freedom ($n = lenght(u_{nodal})$).

Assembly of the global matrices Having defined the connectivity (or assembly) matrix, we can now proceed to assemble the global matrices and vectors.
To do so, we will loop over all the elements and add the contribution of each element to the corresponding global matrix or vector. This concept can be expressed using the gather equation:

$$f_{global} = \sum_{e=1}^{n_{el}} L^{eT} f^e \tag{42}$$

$$M_{global} = \sum_{e=1}^{n_{el}} L^{eT} M^e L^e \tag{43}$$

Where L^e is the connectivity matrix for the $e-th$ element, and f^e and M^e are the elemental force vector and mass matrix, as computed in Equation 30.

4 Implementation

Given that most of the FEM problem is based on the evaluation of integrals, it's now useful to report in the following the implementation the Gaussian quadrature integration method.

```
function result = method_gaussian_quadrature_integration(integrand_func, fem)
    switch fem.N_Gauss_points
    case 1
        weights = [2];
        roots = [0];
    case 2
```

```
        weights = [1, 1];
        roots = [-1/sqrt(3), 1/sqrt(3)];
    case 3
        weights = [5/9, 8/9, 5/9];
        roots = [-sqrt(3/5), 0, sqrt(3/5)];
    otherwise
        error('Order of Gaussian quadrature not implemented (too high)')
    end

    result = 0;
    for point_idx = 1:length(roots)

        result = result + integrand_func(roots(point_idx)) * weights(point_idx);

    end
end
```

The core algorithm of the **explicit integration scheme with Total Lagrangian finite element formulation** is implemented in the 'Main.m' file attached to this report.

5 Results

The numerical results for the problem described in Section 1 are reported in the following.

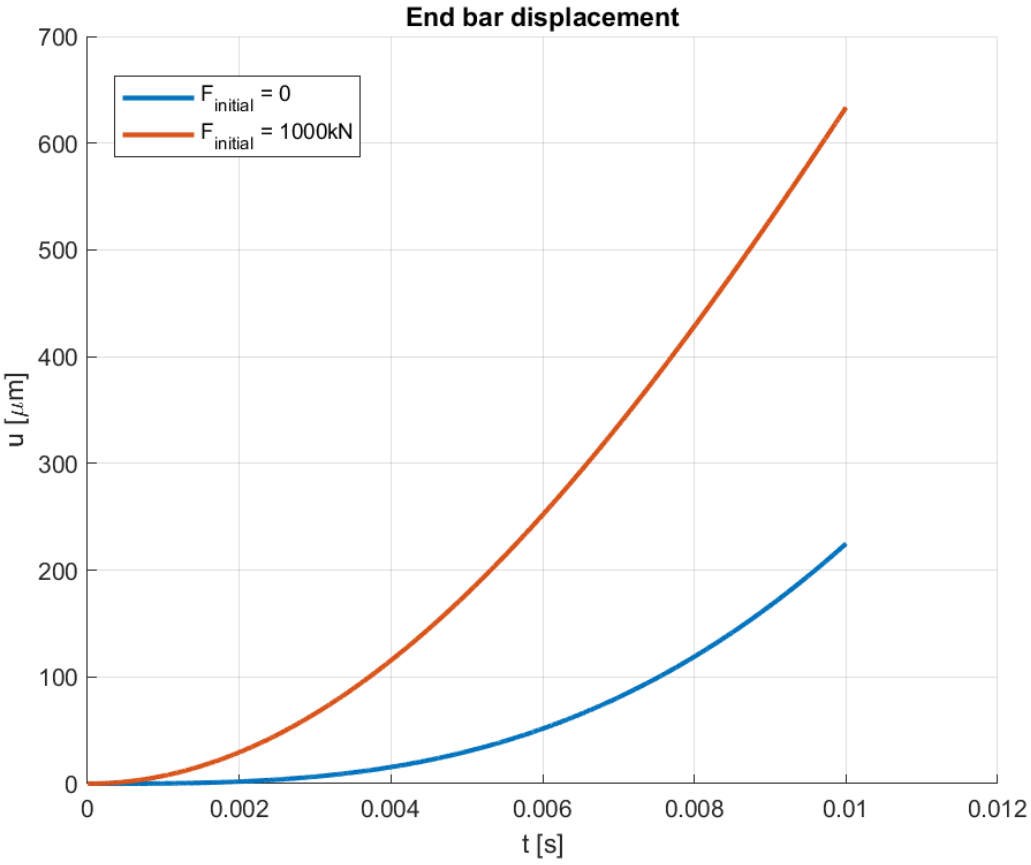

Figure 4: Displacement of the right end of the bar vs. time

In Figure 4 we can observe a comparison between the situation in which the initial load is $F_{initial} = 0$ and linearly increased till $F_{final} = 1000kN$, and the situation in which $F_{initial} = F_{final} = 1000kN$.
In both the situation we can appreciate how the inertia of the bar is able to absorb the initial load.

A Mathematica code

Here follows the `Mathematica` notebook used for symbolic analysis of the equations.

```
Clear["Global`*"]

(*Setting values for the problem datas*)
nElements=1;
nOfNodesPerElement=4;
nOfNodes =(nOfNodesPerElement-1)*nElements + 1 ;

(*Global nodes displachment vector*)
ue=Array[Symbol["ue"<>ToString[#]]&,nOfNodesPerElement];
u=Array[Symbol["u"<>ToString[#]]&,nOfNodes];

ue =ArrayReshape[ue ,{nOfNodesPerElement, 1}];
u =ArrayReshape[u ,{nOfNodes, 1}];

(*Local nodes coordinates*)
points=Array[{-1+#*2/(nOfNodesPerElement-1),0}&,nOfNodesPerElement,0];

x1=0;
x2=x1+le;

X[Xi_]:=(Xi*(x2-x1)+(x1+x2))/2;
Xi[x_]:=(2*x-(x1+x2))/(x2-x1);

(*Local shape functions & compatibility functions*)
(*Notice B0\[Xi] == B0x = D[N0\[Xi]/.\[Xi]->Xi[\[Xi]]/. \[Xi]->x, x];*)
N0\[Xi] = Array[InterpolatingPolynomial[ReplacePart[points,#->{points[[#,1]],1}],\[Xi]]&,
    nOfNodesPerElement];
B0\[Xi] = D[N0\[Xi], \[Xi]]*D[Xi[x], x];

N0\[Xi] =ArrayReshape[N0\[Xi] ,{1, nOfNodesPerElement}];
B0\[Xi] =ArrayReshape[B0\[Xi] ,{1, nOfNodesPerElement}];

(*Local matrices and vectors in differential form*)
dfint[\[Xi]_]:=Transpose[B0\[Xi]]. B0\[Xi] . ue * E0 * A0 * D[X[\[Xi]], \[Xi]];
dfbody[\[Xi]_]:=R0 * A0 * b * Transpose[N0\[Xi]] * D[X[\[Xi]], \[Xi]];
ftraction:=Transpose[N0\[Xi]/.\[Xi]->1]*A0 *\[Tau] - Transpose[N0\[Xi]/.\[Xi]->-1]*A0*\[Tau
    ];
dm[\[Xi]_]:=R0 * A0 * Transpose[N0\[Xi]] . N0\[Xi] * D[X[\[Xi]], \[Xi]];

(*Local matrices and vectors in integral form*)
fint = Integrate[dfint[\[Xi]], {\[Xi], -1, 1}];
fext = Integrate[dfbody[\[Xi]], {\[Xi], -1, 1}] + ftraction;
fkin = fext-fint;
m = Integrate[dm[\[Xi]], {\[Xi], -1, 1}];
```

Listing 1: Mathematica notebook used for symbolic analysis of the equations.