Finite Element Method (2D)

Divyaprakash (2020AMZ8461)

1 Introduction

The force in every node of an element in the Finite Element formulation can be calculated using Equation 1.

$$F_i^\alpha = -\frac{1}{2} c_o A \left[\frac{\partial I_1}{\partial x_i^\alpha} - \frac{2}{J} \left(1 - \frac{K}{c_o} \log J \right) \frac{\partial J}{\partial x_i^\alpha} \right] \tag{1}$$

2 Shape Function Coefficients

The Shape function is given by Equation 2.

$$L_\alpha(\vec{X}) = \frac{A^\alpha(\vec{X})}{A(\vec{X})} \tag{2}$$

Where A is the area of a triangle. It can further simplified into Equation 3.

$$L_\alpha(\vec{X}) = a^\alpha + b_i^\alpha X_i \tag{3}$$

The coefficients a^α and b_i^α can be calculated by calculating the co-factors of the Matrix 4.

$$\begin{bmatrix} x^1 & x^2 & x^3 \\ y^1 & y^2 & y^3 \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

where the co-factors divided by the area of each element correspond to the coefficients (Equation 3) as given in the Matrix 5.

$$\begin{bmatrix} b_1^1 & b_1^2 & b_1^3 \\ b_2^1 & b_2^2 & b_2^3 \\ a_1 & a_2 & a_3 \end{bmatrix} \tag{5}$$

For instance

$$b_1^1 = \frac{1}{A} (-1)^2 \begin{vmatrix} y^2 & y^3 \\ 1 & 1 \end{vmatrix}$$

In MATLAB this is implemented as follows.

1. Generate the 2D Delaunay triangulation
2. Go over each element of the of the triangulation
3. Create a matrix, placing the x and y coordinates of the nodes in the element in the order shown in Matrix 4.
4. Calculate the co-factors of Matrix 4. After diving the values by the area of the element, assign them to the coefficients as specified in Matrix 5.

3 Deformation Gradient Tensor

The deformation gradient tensor is calculated using Equation 6.

$$F_{ij}(t) = \frac{\partial x_i}{\partial X_j} = \sum_{\alpha=1}^{3} b_j^\alpha x_i^\alpha(t) \tag{6}$$

This equation is implemented in MATLAB in form of matrix multiplication as shown in Equation 7. This operation is carried out for each finite element.

$$\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} = \begin{bmatrix} x^1 & x^2 & x^3 \\ y^1 & y^2 & y^3 \end{bmatrix} \begin{bmatrix} b_1^1 & b_2^1 \\ b_1^2 & b_2^2 \\ b_1^3 & b_2^3 \end{bmatrix} \tag{7}$$

4 Jacobian

The Jacobian is caluclated using Equation 8.

$$\mathbf{J} = det(\mathbf{F}) \tag{8}$$

5 Derivative of the First Invariant $\partial I_1/\partial x_i^\alpha$

The derivative of the first invariant is calculated as follows.

$$\begin{aligned} \frac{\partial I_1}{\partial x_k^\beta} &= \frac{\partial}{\partial x_k^\beta}\left(F_{ij}F_{ij}\right) \\ &= 2F_{ij}\frac{\partial F_{ij}}{\partial x_k^\beta} \\ &= 2F_{ij}\sum_{\alpha=1}^{3}\delta_{\alpha\beta}\delta_{ik}b_j^\alpha \\ &= 2F_{kj}b_j^\beta \end{aligned}$$

In MATLAB this is implemented as matrix multiplication for each element.

$$\begin{bmatrix} \frac{\partial I_1}{\partial x_1^1} & \frac{\partial I_1}{\partial x_1^2} & \frac{\partial I_1}{\partial x_1^3} \\ \frac{\partial I_1}{\partial x_2^1} & \frac{\partial I_1}{\partial x_2^2} & \frac{\partial I_1}{\partial x_2^3} \end{bmatrix} = 2\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} b_1^1 & b_1^2 & b_1^3 \\ b_2^1 & b_2^2 & b_2^3 \end{bmatrix} \tag{9}$$

6 Derivative of the Jacobian $\partial J/\partial x_i^\alpha$

The derivative of the Jacobian is calculated as follows.

$$\begin{aligned} \frac{\partial J}{\partial x_m^\beta} &= \epsilon_{ijk}\frac{\partial}{\partial x_m^\beta}\left(F_{1i}F_{2j}F_{3k}\right) \\ &= \epsilon_{ijk}\left[F_{2j}F_{3k}\frac{\partial F_{1i}}{\partial x_m^\beta} + F_{1i}F_{2j}\frac{\partial F_{3k}}{\partial x_m^\beta} + F_{1i}F_{3k}\frac{\partial F_{2j}}{\partial x_m^\beta}\right] \end{aligned}$$

We have the following relation

$$\frac{\partial F_{ij}}{\partial x_m^\beta} = \sum_{\alpha=1}^{3}\delta_{\alpha\beta}\delta_{im}b_j^\alpha$$

Using the above relation we can simplify further.

$$\begin{aligned}\frac{\partial J}{\partial x_m^\beta} &= \epsilon_{ijk}\left[F_{2j}F_{3k}\sum_{\alpha=1}^{3}\delta_{\alpha\beta}\delta_{1m}b_i^\alpha + F_{1i}F_{2j}\sum_{\alpha=1}^{3}\delta_{\alpha\beta}\delta_{3m}b_k^\alpha + F_{1i}F_{3k}\sum_{\alpha=1}^{3}\delta_{\alpha\beta}\delta_{2m}b_j^\alpha\right] \\ &= \epsilon_{ijk}\left[F_{2j}F_{3k}\delta_{1m}b_i^\beta + F_{1i}F_{2j}\delta_{3m}b_k^\beta + F_{1i}F_{3k}\delta_{2m}b_j^\beta\right]\end{aligned}$$

The second term above has a value only when $m = 3$, however in our 2D case m is never 3. Thus it is removed form our calculation.

$$\frac{\partial J}{\partial x_m^\beta} = \epsilon_{ijk}\left[F_{2j}F_{3k}\delta_{1m}b_i^\beta + F_{1i}F_{3k}\delta_{2m}b_j^\beta\right] \tag{10}$$

For the two dimensions we can write Equation 10 separately.

$$\frac{\partial J}{\partial x_1^\beta} = \epsilon_{ijk}F_{2j}F_{3k}b_i^\beta \tag{11}$$

$$\frac{\partial J}{\partial x_2^\beta} = \epsilon_{ijk}F_{1i}F_{3k}b_j^\beta \tag{12}$$

Assuming a plane strain condition gives us, $F_{13} = F_{23} = F_{31} = F_{32} = 0$ and $F_{33} = 1$. Thus in Equation 11-12, $F_{3k} = 1$ only when $k = 3$ and is 0 otherwise. This further reduces the equation as follows.

$$\frac{\partial J}{\partial x_1^\beta} = \epsilon_{ij3}F_{2j}b_i^\beta \tag{13}$$

$$\frac{\partial J}{\partial x_2^\beta} = \epsilon_{ij3}F_{1i}b_j^\beta \tag{14}$$

Out of the all the components of the permutation tensor ϵ_{ij3}, only two have non-zero values, which are, $\epsilon_{123} = 1$ and $\epsilon_{213} = -1$. Thus we have,

$$\frac{\partial J}{\partial x_1^\beta} = \epsilon_{123}F_{22}b_1^\beta + \epsilon_{213}F_{21}b_2^\beta \tag{15}$$

$$\frac{\partial J}{\partial x_2^\beta} = \epsilon_{213}F_{12}b_1^\beta + \epsilon_{123}F_{11}b_2^\beta \tag{16}$$

This calculation can be performed in MATLAB using matrix multiplication as shown in Equation 17.

$$\begin{bmatrix} \frac{\partial J_1}{\partial x_1^1} & \frac{\partial J_1}{\partial x_1^2} & \frac{\partial J_1}{\partial x_1^3} \\ \frac{\partial J_1}{\partial x_2^1} & \frac{\partial J_1}{\partial x_2^2} & \frac{\partial J_1}{\partial x_2^3} \end{bmatrix} = \begin{bmatrix} F_{22} & -F_{21} \\ -F_{12} & F_{11} \end{bmatrix} \begin{bmatrix} b_1^1 & b_1^2 & b_1^3 \\ b_2^1 & b_2^2 & b_2^3 \end{bmatrix} \tag{17}$$

7 Delaunay Triangulation

Vectors containing the x and y coordinates are passed to the Delaunay function. The nodes are numbered according to their index in the input (x and y) vector. The Delaunay function returns a matrix. The rows correspond to each element in the triangulation. The values in each row tell us the index of the x and y vector which make up the vertices of the triangle. In other words, the values in each row give the ordering of the vertices in a triangle in terms of the node/index number of the input x and y vectors.

8 Total force at Each Node

The forces that have been calculated using Equation 1 are stored as 3D matrix, where each page of the matrix is as shown in Matrix 18.

$$\begin{bmatrix} F_1^{1,1} & F_1^{1,2} & F_1^{1,3} \\ F_2^{1,1} & F_2^{1,2} & F_2^{1,3} \end{bmatrix} \tag{18}$$

Each term is denoted as $F_i^{e,n}$, where the superscript e specifies the element (page number). Each row of the matrix, **M** returned by the Delaunay triangulation in MATLAB contains the vertices in an element. The superscript n specifies the column index of that row. The subscript i denotes the force component.

We need to sum up the forces for each node, since the nodes maybe repeated in some elements. We create a vector **FN** which will store the values of force at each node.

In order to sum up the forces, we need to reshape our data such that we know which force corresponds to which node. To do that we reshape our the force matrix as shown in Equation 19. Here the **M** matrix is also reshaped as a vector, with the rows stacked as columns one below the other.

$$\begin{bmatrix} \mathbf{FN}(M(1,1)) \\ \mathbf{FN}(M(1,2)) \\ \mathbf{FN}(M(1,3)) \\ \mathbf{FN}(M(2,1)) \\ \mathbf{FN}(M(2,2)) \\ \mathbf{FN}(M(2,3)) \\ . \\ . \\ . \end{bmatrix} = \begin{bmatrix} F_1^{1,1} & F_1^{1,1} \\ F_1^{1,2} & F_2^{1,2} \\ F_1^{1,3} & F_2^{1,3} \\ F_1^{2,1} & F_2^{2,1} \\ F_1^{2,2} & F_2^{2,2} \\ F_1^{2,3} & F_2^{2,3} \\ . & . \\ . & . \\ . & . \end{bmatrix} \tag{19}$$

Thus the forces at each nodes can be summed by going over the nodes in a loop. The summing up of the x-component force is shown below. Only the first column of the force matrix in Equation 19 is considered.

```
for inode = 1:totalnodes
        FNx(M(inode)) = FNx(M(inode)) + F(inode)
end
```

9 Validation

For the case of plane strain we have,

$$\begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ 2\epsilon_{12} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1-\nu^2 & -\nu(1+\nu) & 0 \\ -\nu(1+\nu) & 1-\nu^2 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{bmatrix} \tag{20}$$

We simulate a case of uni-axial stress along the y-direction. The force, F is applied at the top end of the beam shown in the Figure 1.

Thus, we have $\sigma_{11} = F/A$ and $\sigma_{22} = \sigma_{12} = 0$, where A is the cross-sectional area, $A = b$, since the depth is unity in the z-direction. This leads to the following equations for strain in the x and y directions.

$$\epsilon_{11} = \frac{F}{AE}(1-\nu^2) \tag{21}$$

$$\epsilon_{22} = \frac{F}{AE} - \nu(1+\nu) \tag{22}$$

We then obtain,

$$\Delta l = \frac{Fl}{AE}(1-\nu^2) \tag{23}$$

$$\Delta b = \frac{Fb}{AE} - \nu(1+\nu) \tag{24}$$

The solver is run with the input parameters provided in Table 1. The young's modulus and Poisson's ratio are calculated using equations $E = c_o(2c_0 + 3K)/(c_o + K)$ and $\nu = K/[2(c_0 + K)]$, respectively. The values of the constants in the simulation are, $c_o = 5000$ and $K = 10000$.

Table 1: Simulation Parameters

Properties	Values
Length, l	0.2 m
Breadth, b	0.05 m
Force, F	10 N
Young's Modulus, E	13333.33 N/m^2
Poisson's ratio, ν	0.333

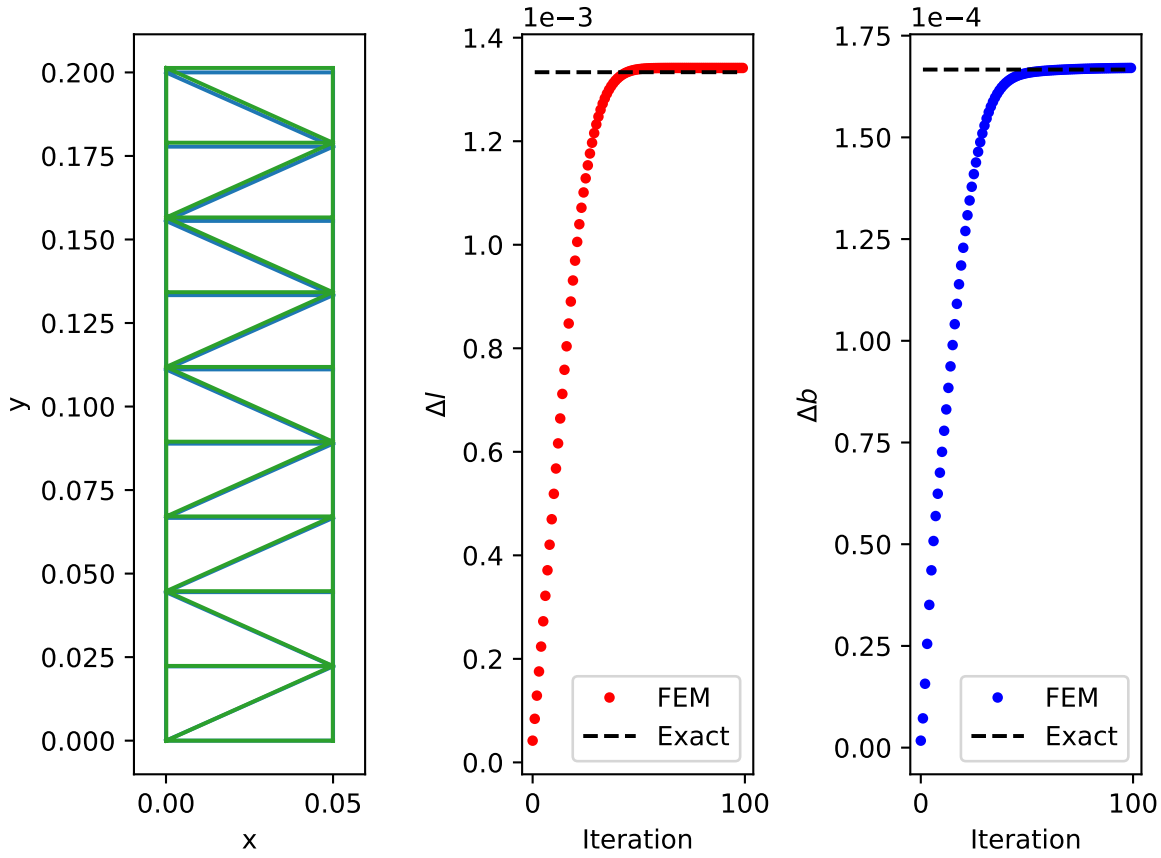

Figure 1: Validation of the FEM model. The dotted black line represents the analytical value of the deformation calculated using Equation 23 & 24

.