Solid bodies displacement simulation using finite element method

Bartosz Bak

June 2024

1 The equation

$$\begin{cases} -\nabla \cdot \sigma = \rho d \\ \overline{\sigma} = C\overline{\varepsilon} \\ u = 0 \text{ over } \Gamma_D \\ \sigma \cdot \overrightarrow{n} = f_n \text{ over } \Gamma_N \end{cases} \tag{1}$$

where

$$\sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \quad \varepsilon = \begin{bmatrix} \varepsilon_{11} & \varepsilon_{12} \\ \varepsilon_{21} & \varepsilon_{22} \end{bmatrix} \quad \overline{\sigma} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ 2\sigma_{12} \end{bmatrix} \quad \overline{\varepsilon} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ 2\varepsilon_{12} \end{bmatrix}$$

$$C = \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}$$

σ and ε are symmetric and will be used as linear operators.

2 The weak form

Expanding the divergence and reshaping it to the weak form

$$\int_\Omega \sum_{i,j=1}^{2} \frac{\delta \sigma_{ij}(u)}{\delta x_i} v_j \; dx = -\int_\Omega \rho d v_j \; dx \tag{2}$$

Using derivative of product and restructuring the equation

$$\int_\Omega \sum_{i,j=1}^{2} \frac{\delta \sigma_{ij}(u) v_j}{\delta x_i} \; dx + \int_\Omega \rho d v \; dx = \int_\Omega \sum_{i,j=1}^{2} \sigma_{ij}(u) \frac{\delta v_j}{\delta x_i} \; dx \tag{3}$$

By applying the law of solid mechanics for strain tensor:

$$\varepsilon(u) = \begin{bmatrix} \frac{\delta u_{11}}{\delta x_1} & \frac{1}{2}(\frac{\delta u_{12}}{\delta x_1} + \frac{\delta u_{21}}{\delta x_2}) \\ \frac{1}{2}(\frac{\delta u_{12}}{\delta x_1} + \frac{\delta u_{21}}{\delta x_2}) & \frac{\delta u_{22}}{\delta x_2} \end{bmatrix} \tag{4}$$

the left hand side may be written as coefficient matrix multiplication

$$\int_\Omega \sum_{i,j=1}^{2} \frac{\delta \sigma_{ij}(u) v_j}{\delta x_i} \; dx + \int_\Omega \rho dv \; dx = \int_\Omega \sigma(u) : \varepsilon(v) \; dx \tag{5}$$

Now let's focus on the right hand side. We can transform by the divergence theorem so that the traction vector can be used in the equation. Note that the third step is possible thanks to the symmetry of σ.

$$\int_\Omega \sum_{i,j=1}^{2} \frac{\delta \sigma_{ij} v_j}{\delta x_i} \; dx = \int_\Omega \sum_{j=1}^{2} \nabla \cdot \sigma_j v_j \; dx = \int_{\delta\Omega} \sum_{j=1}^{2} \sigma_j v_j \cdot \overrightarrow{n} \; dx =$$

$$= \int_{\delta\Omega} \sum_{j=1}^{2} \sigma_j \overrightarrow{n} \cdot v_j \; dx = \int_{\delta\Omega} \overrightarrow{t_n} \cdot v \; dx$$

Finally the equation presents as follows

$$\int_\Omega \sigma(u) : \varepsilon(v) \; dx \; = \int_{\delta\Omega} \overrightarrow{t_n} \cdot v \; dx + \int_\Omega \rho dv \; dx \tag{6}$$

3 Galerkin method

The solution u will be approximated by the sum of coefficients multiplied by the base functions. Each function corresponds to one vertex and each coordinate of a vertex is assigned to one base function. The η_i functions are introduced because the shape of the base functions for both coordinates for given vertex are identical.
The base functions, the solution and the strain operator can be written as

$$v = (\eta_1, 0), (0, \eta_1), (\eta_2, 0), (0, \eta_2) ... (\eta_n, 0), (0, \eta_n) \tag{7}$$

the solution can be written as

$$u = \begin{bmatrix} \sum_{i=1}^{n} x_i \eta_i \\ \sum_{i=1}^{n} y_i \eta_i \end{bmatrix} = \begin{bmatrix} \eta_1 & 0 & \eta_2 & 0 & ... & \eta_n & 0 \\ 0 & \eta_1 & 0 & \eta_2 & ... & 0 & \eta_n \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ ... \\ x_n \\ y_n \end{bmatrix} \tag{8}$$

$$\overline{\varepsilon} = \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{bmatrix} = \begin{bmatrix} \frac{\delta u_1}{\delta x} \\ \frac{\delta u_2}{\delta y} \\ \frac{\delta u_2}{\delta x} + \frac{\delta u_1}{\delta y} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\delta \eta_1}{\delta x} & 0 & \frac{\delta \eta_2}{\delta x} & 0 & ... & \frac{\delta \eta_n}{\delta x} & 0 \\ 0 & \frac{\delta \eta_1}{\delta y} & 0 & \frac{\delta \eta_2}{\delta y} & 0 & ... & \frac{\delta \eta_n}{\delta y} \\ \frac{\delta \eta_1}{\delta y} & \frac{\delta \eta_1}{\delta x} & \frac{\delta \eta_2}{\delta y} & \frac{\delta \eta_2}{\delta x} & ... & \frac{\delta \eta_n}{\delta y} & \frac{\delta \eta_n}{\delta x} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ ... \\ x_n \\ y_n \end{bmatrix} =$$

$$= \begin{bmatrix} \sum_{i=1}^{n} \frac{\delta \eta_i}{\delta x} x_i \\ \sum_{i=1}^{n} \frac{\delta \eta_i}{\delta y} y_i \\ \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta y} x_i + \frac{\delta \eta_i}{\delta x} y_i) \end{bmatrix} dxdy \tag{9}$$

Now using (1) (with translation to operators with dashes), (7) and (9) the equation reduces to a multiplication of a few vectors where u is a unknown.

$$\int_\Omega \overline{\sigma}(v_j) \cdot \overline{\varepsilon}(u)\ dx = \int_\Omega \overline{\varepsilon}(v_j) \cdot C\overline{\varepsilon}(u)\ dx =$$

$$= \frac{E}{1-\nu^2} \int_\Omega \begin{bmatrix} \frac{\delta \eta_j}{\delta x} \\ 0 \\ \frac{\delta \eta_j}{\delta y} \end{bmatrix}^T \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} \sum_{i=1}^{n} \frac{\delta \eta_i}{\delta x} x_i \\ \sum_{i=1}^{n} \frac{\delta \eta_i}{\delta y} y_i \\ \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta y} x_i + \frac{\delta \eta_i}{\delta x} y_i) \end{bmatrix} dxdy =$$

$$= \frac{E}{1-\nu^2} \int_\Omega \begin{bmatrix} \frac{\delta \eta_j}{\delta x} \\ 0 \\ \frac{\delta \eta_j}{\delta y} \end{bmatrix}^T \begin{bmatrix} \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta x} x_i + \nu \frac{\delta \eta_i}{\delta y} y_i) \\ \sum_{i=1}^{n} (\nu \frac{\delta \eta_i}{\delta x} x_i + \frac{\delta \eta_i}{\delta y} y_i) \\ \frac{1-\nu}{2} \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta y} x_i + \frac{\delta \eta_i}{\delta x} y_i) \end{bmatrix} dxdy =$$

$$= \frac{E}{1-\nu^2} \int_\Omega \frac{\delta \eta_j}{\delta x} \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta x} x_i + \nu \frac{\delta \eta_i}{\delta y} y_i) + \frac{\delta \eta_j}{\delta y} \frac{1-\nu}{2} \sum_{i=1}^{n} (\frac{\delta \eta_i}{\delta y} x_i + \frac{\delta \eta_i}{\delta x} y_i) dxdy =$$

$$= \frac{E}{1-\nu^2} \sum_{i=1}^{n} (x_i (\int_\Omega \frac{\delta \eta_i}{\delta x} \frac{\delta \eta_j}{\delta x} dxdy\ + \frac{1-\nu}{2} \int_\Omega \frac{\delta \eta_i}{\delta y} \frac{\delta \eta_j}{\delta y} dxdy\ ) +$$

$$+ (y_i (\nu \int_\Omega \frac{\delta \eta_i}{\delta y} \frac{\delta \eta_j}{\delta x} dxdy\ + \frac{1-\nu}{2} \int_\Omega \frac{\delta \eta_i}{\delta x} \frac{\delta \eta_j}{\delta y} dxdy\ ))$$

For displacement in the y direction the formula can be derived similarly

$$\frac{E}{1-\nu^2} \sum_{i=1}^{n} (x_i (\int_\Omega \frac{\delta \eta_i}{\delta y} \frac{\delta \eta_j}{\delta y} dxdy\ + \frac{1-\nu}{2} \int_\Omega \frac{\delta \eta_i}{\delta x} \frac{\delta \eta_j}{\delta x} dxdy\ ) +$$

$$+ (y_i (\nu \int_\Omega \frac{\delta \eta_i}{\delta x} \frac{\delta \eta_j}{\delta y} dxdy\ + \frac{1-\nu}{2} \int_\Omega \frac{\delta \eta_i}{\delta y} \frac{\delta \eta_j}{\delta x} dxdy\ ))$$

4 The base functions

The base functions for triangle elements are pyramids which value is equal to 1 at their corresponding node and equal to 0 at other nodes. They belong to $\mathcal{C}^2(\Omega)$ function space.
Their formula can be written as

$$\eta_i(x_1, x_2) = \frac{1}{2\,|e_S|} \begin{vmatrix} x_1 - x_{1,i+1} & x_2 - x_{2,i+1} \\ x_1 - x_{1,i+2} & x_2 - x_{2,i+2} \end{vmatrix} \tag{10}$$

and the derivative

$$\frac{\delta \eta_i}{\delta x_p} = (-1)^p \frac{x_{p+1,i+1} - x_{p+1,i+2}}{2\,|e_S|} \tag{11}$$

where $|e_S|$ is the area of the triangle element

$$e_S = \begin{vmatrix} x_{1,i} - x_{1,i+1} & x_{2,i} - x_{2,i+1} \\ x_{1,i} - x_{1,i+2} & x_{2,i} - x_{2,i+2} \end{vmatrix}$$

5 The stiffness matrix assemblation

For a given triangle element e_S which has three local vertices ($i = 1, 2, 3$) there is a local stiffness matrix generated that is translated to the global indexing in is assembled into rows and columns which correspond to correct base functions. So the program will loop over the elements and calculate the integrals by splitting the domain into parts based on the mesh definition.
The program will need to calculate the integral of two base function derivatives multiplied by each other. By applying (11) and taking into consideration the fact that the function over the integral is constant the integral presents as follows

$$\int_{e_S} \frac{\delta \eta_i}{\delta x_p} \frac{\delta \eta_j}{\delta x_q} dx_1 dx_2 = (-1)^{p+q} \frac{(x_{p+1,i+1} - x_{p+1,i+2})(x_{q+1,j+1} - x_{q+1,j+2})}{4\,|e_S|} \tag{12}$$

Additionally, to calculate the nodal forces it is required to calculate the integral of the base function η_i over an element. It can be easily obtained by a geometric observation. The height of the pyramid over an element is equal to 1 and the area of it is known at this point. Using the formula for volume of a pyramid there is

$$\int_{e_S} \eta_i \; dx_1 dx_2 = \frac{1}{3}\,|e_S| \tag{13}$$

6 Implementation

In the *main.cpp* the program assembles the stiffness matrix and assigns proper values to the nodal forces vector. The MshMeshLoader reads a .msh file given in the first parameter from the command line and writes the mesh into the memory. The msh format is used by GMSH software to construct various meshes for FEM calculations https://gmsh.info/. The nodes (*Node* class) and elements (*Element*

class) are stored in the Mesh class. The *Element* has an array of size three which maps local indices into global ones. In the *TriangleElementWrapper.cpp* helping functions for integral calculations are implemented. The main file also imports constants from the file which name was given in the fourth parameter.

7 Neuman condition

The class *NeumanCondition* handles the Neuman condition data. It loads the file by its name given as the third parameter from the command line. The file should start with line **BARTA_NEUMAN** following by a line with a positive number indicating number of elements affected by Nueman condition. For each element there goes one line with firstly the label of the element (assigned by GMSH in the .msh file) and the number (1, 2 or 3) of edges affected by the Neuman condition. For each edge there are three numbers in the line - the first is the label of the starting node (which is its index + 1) of the edge. The program assumes that the edges of the triangle are directed counterclockwise. The second and the third numbers are the forces in x and y direction respectively. Note that these are not the coefficients of the traction vector.

8 Dirichlet condition

The class *DirichletSolver* handles the Dirichlet condition data. It loads the file by its name given as the second parameter from the command line. The file should start with line **BARTA_DIRICHLET** following by a line with a positive number indicating the count of nodes affected by Dirichlet condition. Then for each node in a separate line there is one number which is interpreted as the label of the node. The program inserts row consisting of zeros and one 1 into the stiffness matrix and places 0 into the nodal forces vector.

9 The result

After calculation the result is written into result.msh file which can be viewed in the GMSH GUI. In the examples directory there are some example data files which can be used by the program. The below example shows a chart of the deformed mesh from the omega example.

Figure 1: Initial mesh

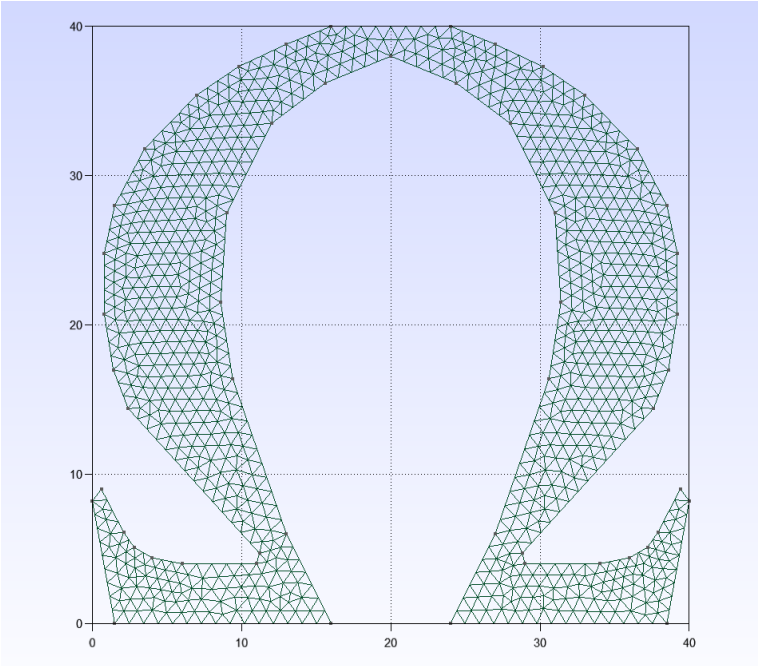

Figure 2: Deformed mesh