Programming project in TMA4220, part 2C:
Structural analysis with the equations of linear elasticity

October 22, 2015

In this project you will look into how structures deform under loads and study the (elastic) forces acting on a body using the Linear Elasticity model. Since the model is linear, it leads to easy and efficient computations, but also has some important drawbacks, which hopefully you might be able to see. After coding a finite element solver for the Linear Elasticity equation, you will apply it to a mesh of your choice to produce plots or animations of how the mesh is deformed and/or of how the internal stresses are distributed.

1 Introduction

We are in this problem going to consider the linear elasticity equation. The equations describe deformation and motion in a continuum. While the entire theory of continuum mechanics is an entire course by itself, it will here be sufficient to only study a small part of this: the linear elasticity. This is governed by three main variables u, ε and σ:

$u = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$ The *displacement* vector measures how much each spatial point has moved in (x, y)-direction.

$\varepsilon = \begin{bmatrix} \varepsilon_{xx} & \varepsilon_{xy} \\ \varepsilon_{xy} & \varepsilon_{yy} \end{bmatrix}$ The *strain* tensor measures how much each spatial point has deformed or stretched.

$\sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}$ The *stress* tensor measures the forces per area acting on a particular spatial point.

We will describe all equations and theory in terms of two spatial variables (x, y), but the extension into 3D should be straightforward. Note that the subscript denotes vector component and *not* derivative, i.e. $u_x \neq \frac{\partial u}{\partial x}$.

The three variables u, ε, σ depend on one another in the following way:

$$u = u(x) \tag{1}$$

$$\varepsilon = \varepsilon(u) \tag{2}$$

$$\sigma = \sigma(\varepsilon) \tag{3}$$

The primary unknown u (the displacement) is the one we are going to find in our finite element implementation. From (1) we will have two displacement values for each finite element "node", one in each of the spatial directions.

The relation (2) is a purely geometric one. Consider an infinitesimal small square of size dx and dy, and its deformed geometry as depicted in Figure 1. The strain is defined as the stretching of the element, i.e. $\varepsilon_{xx} = \frac{\text{length}(ab) - \text{length}(AB)}{\text{length}(AB)}$.

The complete derivations of these quantities is described well in the Wikipedia article on strain, and the result is

$$\varepsilon = \frac{1}{2}\left(\nabla u + (\nabla u)^T\right)$$

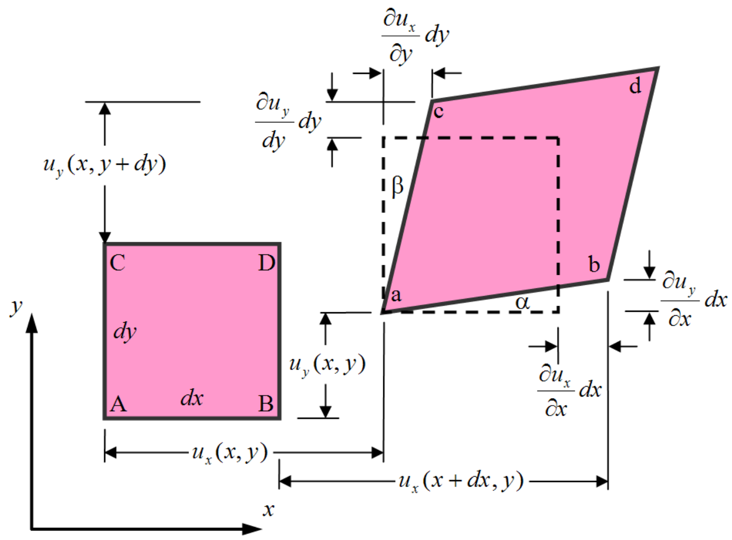

Figure 1: An infinitesimal small deformed rectangle.

or written in component form,

$$\varepsilon_{xx}(u) = \frac{\partial u_x}{\partial x}$$
$$\varepsilon_{yy}(u) = \frac{\partial u_y}{\partial y}$$
$$\varepsilon_{xy}(u) = \varepsilon_{yx}(u) = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}.$$

Note that these relations are the *linearized* quantities, which will only be true for small deformations.

For the final relation, which connects the deformation to the forces acting upon it, we turn to the material properties. Again, there is a rich literature on the subject, and different relations or physical laws to describe different materials. In our case, we will study small deformations on solid materials like metal, wood or concrete. It is observed that such materials behave elastically when under stress of a certain limit, i.e. a deformed geometry will return to its initial state if all external forces are removed. Experiment has shown that the Generalized Hooke's Law is proving remarkable accurate under such conditions.

Consider a body being dragged to each side by some stress σ_{xx} as depicted in Figure 2. Hooke's law states that the forces on a spring depends linearly on the amount of stretching multiplied by some stiffness constant, i.e. $\sigma_{xx} = E\varepsilon_{xx}$. The constant E is called Young's modulus. Generalizing upon this law, we see that materials typically contract in the y-direction when dragged in the x-direction, and *vice versa*. The ratio of compression vs expansion is called Poisson's ratio ν and is expressed as $\varepsilon_{yy} = -\nu\varepsilon_{xx}$. This gives the following relations:

$$\varepsilon_{xx} = \frac{1}{E}\sigma_{xx}$$
$$\varepsilon_{yy} = -\frac{\nu}{E}\sigma_{xx}$$

Due to symmetry conditions, we clearly see that when applying a stress σ_{yy} in addition to σ_{xx} we get

$$\varepsilon_{xx} = \frac{1}{E}\sigma_{xx} - \frac{\nu}{E}\sigma_{yy}$$
$$\varepsilon_{yy} = \frac{1}{E}\sigma_{yy} - \frac{\nu}{E}\sigma_{xx}$$

Finally, it can be shown that the relation between the shear strain and shear stress is $\varepsilon_{xy} = 2\frac{1+\nu}{E}\sigma_{xy}$. We end up with the relation

$$\bar{\sigma} = C\bar{\varepsilon} \tag{4}$$

Figure 2: Deformed geometry under axial stresses.

where $\bar{\sigma}$, $\bar{\varepsilon}$ and C are defined as

$$\bar{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{xy} \end{bmatrix}, \qquad \bar{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \qquad C = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}, \qquad C^{-1} = \begin{bmatrix} \frac{1}{E} & -\frac{\nu}{E} & 0 \\ -\frac{\nu}{E} & \frac{1}{E} & 0 \\ 0 & 0 & 2\frac{1+\nu}{E} \end{bmatrix}.$$

For a body at static equilibrium, we have the governing equation

$$\begin{cases} \nabla \cdot \sigma(u) = -f & \text{in } \Omega \\ u = g & \text{on } \Gamma_D \\ \sigma(u) \cdot n = h & \text{on } \Gamma_N \end{cases} \tag{5}$$

where

- n is the outward unit normal on $\partial\Omega$
- $\Gamma_D, \Gamma_N \subset \partial\Omega$ are disjoint sections of the boundary of Ω such that $\partial\Omega = \Gamma_D \cup \Gamma_N$
- $f \in L^2(\Omega)^2$ is a (vector-valued) force acting on the body
- g, h are given boundary data.

By the divergence of a matrix-valued quantity we mean row-wise differentiation:

$$\nabla \cdot \sigma = \nabla \cdot \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x}\sigma_{xx} + \frac{\partial}{\partial y}\sigma_{xy} \\ \frac{\partial}{\partial x}\sigma_{xy} + \frac{\partial}{\partial y}\sigma_{yy} \end{bmatrix}.$$

2 Project description

NOTE: This is a *suggestion* for a project description. You will be graded based on what you do and how you do it, NOT on how many of the points below you have completed. However, at a *minimum* you need to derive the weak formulation of the stationary problem, implement a 2D or 3D (or both) finite element code and test your code by comparing with an exact solution.

1. Let $g \equiv 0$. By multiplying with a test function $\mathrm{v} = \begin{bmatrix} \mathrm{v}_1(x,y) \\ \mathrm{v}_2(x,y) \end{bmatrix}$ and integrating over the domain Ω, show that (5) can be written as the scalar equation

$$\sum_{i=1}^{2}\sum_{j=i}^{2}\int_{\Omega}\varepsilon_{ij}(\mathrm{v})\sigma_{ij}(u)\,dA = \sum_{i=1}^{2}\int_{\Omega}\mathrm{v}_i f_i\,dA + \sum_{i=1}^{2}\int_{\Gamma_N}\mathrm{v}_i h_i\,dS$$

(where we have exchanged the subscripts (x, y) with $(1, 2)$). Show that this can be written in compact vector form as

$$\int_{\Omega}\bar{\varepsilon}(\mathrm{v})^T C\bar{\varepsilon}(u)\,dA = \int_{\Omega}\mathrm{v}^T f\,dA + \int_{\Gamma_N}\mathrm{v}^T h\,dS. \tag{6}$$

2. Let v be a test function in the space X_h^1 of piecewise linear functions on some triangulation $\mathcal{T}_h$. Note that unlike before, we now have *vector* test functions. This means that for each node $\hat{i}$, we will have two test functions

$$\varphi_{\hat{i},1}(x) = \begin{bmatrix} \varphi_{\hat{i}}(x) \\ 0 \end{bmatrix}, \qquad \varphi_{\hat{i},2}(x) = \begin{bmatrix} 0 \\ \varphi_{\hat{i}}(x) \end{bmatrix}.$$

Let these functions be numbered by a single running index $i = 2\hat{i} + j$, where i is the node number in the triangulation and j is the vector component of the function.

Show that by inserting $\mathrm{v} = \varphi_j$ and $u = \sum_i \varphi_i u_i$ into (6) you get the system of linear equations

$$Au = b$$

where

$$A_{i,j} = \int_{\Omega}\bar{\varepsilon}(\varphi_i)^T C\bar{\varepsilon}(\varphi_j)\,dA, \qquad b_i = \int_{\Omega}\varphi_i^T f\,dA + \int_{\Gamma_N}\varphi_i^T h\,dS.$$

3. Show that

$$u = \begin{bmatrix} (x^2-1)(y^2-1) \\ (x^2-1)(y^2-1) \end{bmatrix}$$

is a solution to the problem

$$\begin{cases} \nabla\cdot\sigma(u) = -f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{7}$$

when

$$f_x = \frac{E}{1-\nu^2}\left(-2y^2 - x^2 + \nu x^2 - 2\nu xy - 2xy + 3 - \nu\right)$$
$$f_y = \frac{E}{1-\nu^2}\left(-2x^2 - y^2 + \nu y^2 - 2\nu xy - 2xy + 3 - \nu\right)$$

and $\Omega = (-1, 1)^2$.

4. Modify your Poisson solver to solve the problem (7). Verify that you are getting the correct result by comparing with the exact solution. The mesh may be obtained through the Grid function `getPlate( )`.

5. Modify your 3D Poisson solver to assemble the stiffness matrix from linear elasticity in three dimensions.

Import a 3D mesh from Minecraft (read the instructions on how to do this in the previous years webpages of the course) or create a mesh using your choice of mesh generator. Apply gravity loads as the body forces acting on your domain, this will be the right hand side function f in (5). In order to get a non-singular stiffness matrix you will need to pose some Dirichlet boundary conditions. Typically you should introduce zero displacements (homogeneous Dirichlet conditions) where your structure is attached to the ground. This would yield a stationary solution.

3 Further ideas

- Solving (5) with a finite element method gives you the primary unknown: the displacement u. If you are interested in derived quantities such as the stresses, these can be calculated from (4). Note that σ is the (symmetric) gradient of u which means that since u is C^0 across element boundaries, then σ will be discontinuous. To get stresses at the nodal values, we propose to average the stresses over all neighbouring elements.

  Loop over all elements and evaluate (the constant) stresses on that element. For each node, assign the stresses to be the average stress over all neighbouring elements. This method is called "Stress Recovery".

- Implement and test the time-dependent problem

$$\rho \frac{\partial^2 u}{\partial t^2} = \nabla \cdot \sigma(u) + f. \tag{8}$$

  Here, $\rho > 0$ is the mass density of the material.

- You can set an upper limit for the stresses on any given element (according to the structural properties of the material under consideration). Then, if and when the stress on a particular element exceeds the limit, you consider that part of the material as broken, and "deactivate" the element, removing it from your mesh. How does this affect the load distribution in your structure? Will it lead to a collapse?