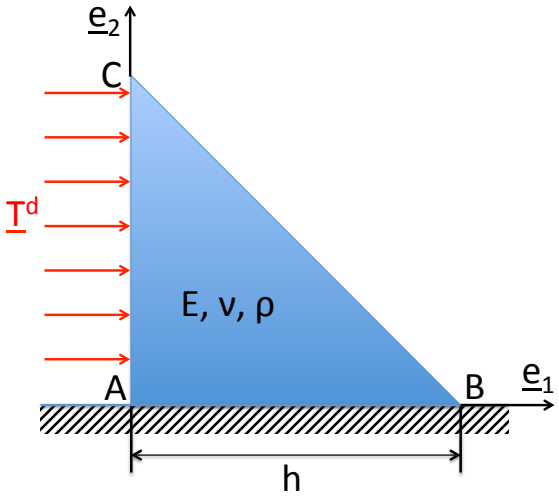
PW1 - Estimation of the displacement at the top of a gravity dam thanks to various modeling techniques.

1 Introduction

1.1 Problem Data

This study is about about a prism-shaped gravity dam with a triangular section. The longest dimension of the dam is parallel to the third vector of the cartesian spatial basis $(\underline{e}_1, \underline{e}_2, \underline{e}_3)$.
The length L of the dam is considered very high compared to the dimensions of the section. The plane strains hypothesis can therefore be assumed to be true. The section ABC defined in the plane orthogonal to $\underline{e}_3$ and represented on the figure besides is consequently now considered. This section is more precisely an isosceles right-angled triangle. The identical sides of the triangle are $AB = AC = h = 15\,m$.



Schematic representation of the dam and its load.

The constitutive material (concrete) is supposed to be linear elastic isotropic and homogeneous. Its characteristics are reported below:

- Young modulus: $E = 35$ GPa
- Poisson ratio: $\nu = 0.2$
- Density: $\rho_c = 2300\,\mathrm{kg.m^{-3}}$

The basis AB of the dam is embedded and in a first case scenario the action of gravity is neglected. Loads are therefore limited to the sole pressure of water (side = AC). The pressure is supposed to be uniformly distributed on the upstream face of the dam (which is a very strong hypothesis). The surface force applied to AC is therefore:

$$\underline{T}^d = p_0 \underline{e}_1 \quad \text{with} \quad p_0 = p_{\text{atm}} + \frac{1}{2}\rho_w g h,$$

where $p_{\text{atm}} = 10^5$Pa is the atmospheric pressure and $\rho_w = 1000\,\mathrm{kg.m^{-3}}$ the water density. Consequently the resultant of the linear force density applied in 2D is equal to the resultant of the hydrostatic pressure (see Section 6).

The main goal of this study is to estimate the displacement at the top of the dam (point C) when loaded, thanks to finite element analysis. Two types of elements will be considered : linear elastic elements (T3) and quadratic elements (T6). The results obtained using both modeling will then be compared and analysed.

1.2 Modeling tools

It is reminded that the pre- and post-processing phases can be performed by using GMSH (mesh generator) provided by Christophe Geuzaine and Jean-François Remacle `http://gmsh.info/`
Problem solving will be achieved thanks to the -so far- 2D finite element code provided by Jeremy Bleyer, available on the Moodle platform.

2 Study with T3 elements

1. **Pre-processing**. Write the script describing the geometry of the problem (file *Geo_ Dam.geo*). Choose different characteristic lengths (lc1, lc2,...) in order to refine the mesh where stresses should be the highest a priori.

2. **Solving (Main script)**. Create a new script called *Gravity_ Dam.py*. Fill in the heading (date, author, short description,...). Then import the *wombat* library containing the functions of the code.
   In the same file write a script allowing to solve the problem using the finite element library. You may get inspired by the script used to solve the wing problem *Wing.py*. Note that at this step the mesh will be generated using linear triangular elements (T3).

3. **Post-processing**

   - Extract the values corresponding to the displacement components at point C where the displacement is maximum.
   - Extract the values corresponding to the maximum Von Mises equivalent stress at point A (bottom corner).

4. **Influence of mesh size**.

   - Try again while refining the mesh at each attempt (up to 30,000 nodes approximately). Plot the evolution of the displacement at the top with respect to the number of elements in the mesh. Conclude.
   - On another graph plot the values of maximum Von Mises equivalent stress versus the number of elements in the model.
   - Are the results obtained expected ? Analyse the results and suggest a solution to improve them.

   Tip: Here is a syntax to plot curves in Python.

```
plt.plot(ValX,ValY,'-bo',Val2X,Val2Y,'-rx')
plt.xlabel('blabla')
plt.ylabel('blibli')
plt.legend(('ValY','Val2Y'))
plt.show()
```

3 Study with T6 elements

T6 elements are not fully implemented in the FE library provided. In order to perform simulations on the dam using T6 elements a certain number of functions require to be coded allowing to compute the stiffness matrix and estimate the stress tensor components.

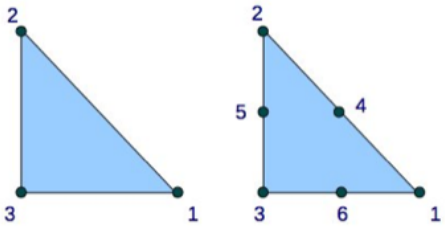
5. **Element definition** In the directory called *element*, open the script *solidT6.py*. Complete the class functions called `shape_functions`, `compute_Be_matrix`, `elementary_stiffness` and `stresses`.

   Tip: See *Help_ stresses.pdf*. Inspiration may come from the file *solidT3.py* and for the already implemented methods in *solidT6.py*

6. **Pressure action** Perform a structural calculation using a new script *Gravity_ Dam_ T6.py*, in order to simulate the response of the dam submitted to water pressure p_0. Extract the values corresponding to the displacement components at point C on top of the dam and compare them to the values obtained with a T3 mesh for approximately the same number of elements.

   Tip: The .geo file which only contains geometrical features concerning the dam does not need to be changed. But to use T6 elements one should modify the following line :
   `mesh, boundary = call_gmsh('geometry/Geo_Dam.geo',SolidT6)`

   Note: The reference configuration and shape functions of the elements used in the FE code provided are given below. Verify that $N_i(x_j) = \delta_{ij}$.



T3 and T6 reference elements

For T3:
$N_1 = a_1$
$N_2 = a_2$
$N_3 = 1 - a_1 - a_2$

For T6:
$N_1 = a_1(2a_1 - 1)$
$N_2 = a_2(2a_2 - 1)$
$N_3 = (1 - a_1 - a_2)$
$\times(1 - 2a_1 - 2a_2)$
$N_4 = 4a_1a_2$
$N_5 = 4a_2(1 - a_1 - a_2)$
$N_6 = 4a_1(1 - a_1 - a_2)$

4 Gravity action

7. **Volume forces** Complete the code allowing to take into account the contribution of volume force densities (such as gravity effects) to the right hand side term $\{F\}$.

   Tip: See *Help_ stresses.pdf*. The element libraries *SolidT3.py* and *SolidT6.py* must be enriched, as well as the assembly function `assembl_external_forces` in *finite_ elements_ sparse.py*.

5 Energy estimate

8. **Post-processing** In the post-processing part of the code dealing with the results *res_ treat.py*, write a new function allowing to estimate the value of potential energy E_{ph} easily in this specific problem.
   The expression of the potential energy is reminded here.

$$E_{ph} = \frac{1}{2}\int_{\Omega^h} \underline{\underline{\epsilon}}\,(\underline{u}_h) : \mathbf{A} : \underline{\underline{\epsilon}}\,(\underline{u}_h)\, dV - \int_{\Omega^h} \underline{f}_v.\underline{u}_h dV - \int_{\partial\Omega_F} \underline{T}^d.\underline{u}_h dS \tag{1}$$

   Tip: If `numpy` is imported as `np`, then products between matrices and vectors can be obtained this way: is `M` is a matrix, `MS` a *sparse* matrix and `(V,W)` two vectors,

```
M.V → np.dot(M,V)
MS.V → MS.dot(V)
V.W → np.inner(V,W)
```

9. **Convergence study** For T3 and T6 elements, study the convergence rate of potential energy with respect to the refinement of the mesh.

6 [BONUS] Heterogeneous hydrostatic pressure

This section is optional and will provide bonus points if (correctly) adressed.

The assumption that the water pressure is homogeneous simplifies the computations, but is inaccurate: the pressure increases linearly with the depth as

$$\underline{T}^d(x_2) = p(x_2)\underline{e}_1 \quad \text{with} \quad p(x_2) = p_{\text{atm}} + \rho_w g(h - x_2), \qquad x_2 \in [0, h].$$

10. Create new functions (e.g. `add_heterogeneous_distributed_forces` in *forces.py*) and adapt the code to account for heterogenous surface forces for T3 elements.

    Tip: The elementary contribution of surface load comes from integrals on *boundary elements* (or "trace" elements). Therefore modifications (or new methods) must also be added in *trace_ elements.py.*

    Tip: The force needs to be evaluated at Gauss points of the boundary elements (2 Gauss points for `TraceT3` elements), during the elementary force vector computation. This evaluation can be done (i) a priori (in the main file) for all Gauss points, but then arrays of values must be passed as arguments of the various functions (`(fx,fy)` will be arrays instead of scalars, and must be splitted over all elements), or (ii) when needed, inside each element, by passing *functions* (e.g. $x_2 \mapsto \underline{T}^d(x_2)$) as arguments.

11. Run again the code for this new (more realistic) load, and discuss the differences with the previous case (e.g. on the maximal displacement and/or maximal VM stress and/or potential energy).

12. Repeat the previous questions for T6 elements.