

Practical Work: Coding the Penalty Contact Element in a toy-FEM code

Vladislav A. Yastrebov

February 2025

1. Introduction

In this practical work, we will code the penalty contact element in a toy-FEM code. The hardcoded truss element and a rigid wall obstacle are shown in the figure below. The boundary conditions are as follows: + Two left nodes are clamped (displacements are fixed $U_x = U_y = 0$) + On the middle node, a vertical force F_y , pointing downwards, is applied.

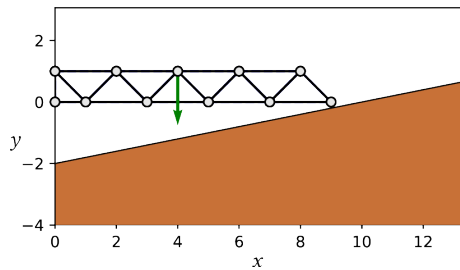


Figure 1: Truss element and rigid wall obstacle

Our task is to code the penalty contact element to prevent the structure from penetrating the rigid wall.

2. Step-by-step instruction

1. Check out the definition of the rigid obstacle (plane wall):

```
# Rigid wall (obstacle) geometry
# outward normal vector
normal = np.array([-1,5])
normal = normal/np.linalg.norm(normal)

tangent = normal.copy()
tangent[0] = normal[1]
```

```

tangent[1] = -normal[0]

# y = y_wall + tangent[1]/tangent[0] * (x-x_wall)
x_wall = 5
y_wall = -1

```

The geometry is given by a single point $\mathbf{x}_w = \{x_{wall}, y_{wall}\} = \{5, -1\}$ and a normal vector $\mathbf{n} = \{n_x, n_y\} = \{-1, 5\}$, the tangent vector is $\mathbf{t} = \{t_x, t_y\} = \{5, 1\}$.

2. Now we need to find normal projections for all nodes of the truss element onto the rigid wall, check the following code:

```

# FIXME: code the projection of every node on the rigid wall
proj_x = 0 # ?? to be defined
proj_y = 0 # ?? to be defined
plt.plot(proj_x, proj_y, "v", color="r")

```

It is for you to code the projection of every node on the rigid wall. We recall that the point coordinates in the current configuration are given by variables x_i and y_i for the i -th node.

3. Define the gap function as a difference between the node coordinates and its projection on the rigid wall:

```

# define the gap = (xi-proj).normal
# FIXME: code the gap expression
gap = 0 # ?? to be defined

```

1. Now, we will code the residual vector. For the penalty contact element, the residual vector is given by:

$$[R([u])] = [R^S([u])] + [R^C([u])],$$

where $[R^S([u])]$ is the residual vector from the solid mechanics problem (truss element) and $[R^C([u])]$ is the residual vector from the contact problem (penalty contact element). The contact residual vector for a single node is given by:

$$R_i^C = \begin{cases} \epsilon_n |g_n| \mathbf{n}, & \text{if } g_n < 0, \\ 0, & \text{if } g_n \geq 0, \end{cases}$$

where g_n is the normal gap function (that we have already defined as **gap**), ϵ_n is the penalty parameter, and \mathbf{n} is the normal vector to the i -th element. The normal vector comes from the fact that for a single element, the variation of the normal gap is given by

$$\delta g_n = \delta(\mathbf{x} - \mathbf{x}_w) \cdot \mathbf{n} = \delta \mathbf{x} \cdot \mathbf{n},$$

where $\delta \mathbf{x}$ is the variation of the node coordinates and \mathbf{x}_w is the projection of the node on the rigid wall. If there's no contact, the associated add-on from the contact problem is zero. You will need to encode this expression in the following code:

```
# FIXME: code the residual vector
F[ii] += 0 # ?? to be defined
F[jj] += 0 # ?? to be defined
```

where `ii` and `jj` are the global indices of the DOFs for the i -th node.

2. Now, we will code the tangent matrix. For the penalty contact element, the tangent matrix is given by:

$$[K([u])] = [K^S([u])] + [K^C([u])],$$

where $[K^S([u])]$ is the tangent matrix from the solid mechanics problem (truss element) and $[K^C([u])]$ is the tangent matrix from the contact problem (penalty contact element). The contact tangent matrix for a single node is given by:

$$[K_i^C] = \frac{\partial R_i^C}{\partial \delta u_i} = \begin{cases} \epsilon_n \mathbf{n} \otimes \mathbf{n}, & \text{if } g_n < 0, \\ 0, & \text{if } g_n \geq 0, \end{cases}$$

where \otimes is the tensor product. You will need to encode this expression in the following code:

```
# FIXME: code the tangent matrix
mesh.K[ii][ii] += 0 # ?? to be defined
mesh.K[ii][ii+1] += 0 # ?? to be defined
mesh.K[ii+1][ii] += 0 # ?? to be defined
mesh.K[ii+1][ii+1] += 0 # ?? to be defined
```

where `ii` and `jj` are the global indices of the DOFs for the i -th node.

3. Effect of the penalty parameter

Now, as soon as the code is working, we can play with the penalty parameter, defined as `penalty` in the code. Check how the topology of the contact and the resulting penetration are affected by the penalty parameter.

4. Solution

The solution is given in the file `solved_contact.py`.