University of Liège
Faculty of Applied Sciences

Implementation of a FEM-BEM Solver for Electrostatic Actuation

Romain Boman
Christophe Geuzaine

Louis Denis
Pierre-Yves Goffin
Kévin Maltez
Valentin Vanraes

MATH0471
Multiphysics Integrated Computational Project

May 2022

Contents

Introduction

In this report, the implementation of a two-dimensional coupled elasto-electrostatic FEM-BEM solver is described. Such a solver allows to tackle problems involving electrostatic actuation which are particularly useful in the realm of microsystems. The presented program implements the Finite Element Method (FEM) for solving the elastic problem and the Boundary Element Method (BEM) for solving the electrostatic problem.

First, both methods are presented and the corresponding theoretical considerations are introduced. After explaining the details of the numerical implementation, each method is validated against reference solutions. The numerical behaviour of both methods is studied in detail. The convergence or the computation time, among many others, are explored and described extensively.

In Section 3, the one-way coupling method between the BEM solver and the FEM solver is explained. The fundamental concept of electrostatic pressure is derived and introduced, before providing more information about the communication between both solvers. Finally, the coupling is once again validated in a simple reference configuration.

Such a linear solver is interesting for understanding the physical mechanisms related to electrostatic actuation. However, it relies on the assumption of small displacements, which is not always satisfied in the context of microsystems (even if the displacements are small, by definition, in microsystems, they can be important relative to the dimensions of the considered problem). To tackle more realistic problems involving large displacements, a non-linear FEM solver has been implemented. The theory related to the so-called *corotational approach*, which allows to handle large rotations, is introduced in Section 4. Moreover, the corresponding Newton-Raphson iterative algorithm and the numerical implementation of the method is described in detail. The non-linear elastic solver is validated against results found in the literature, before quantifying the performance and the convergence of the code.

Once such a non-linear solver is implemented, it allows to develop a two-way coupled iterative FEM-BEM solver, in which both individual solvers communicate until an equilibrium configuration is reached. It allows to retrieve the physical distribution of the coupled elasto-electrostatic fields, as the displacement of a given structure also influences the electric field around that given structure. The implementation of such an iterative solver is explained and discussed in Section 5.

Finally, in the last section, the coupled solver is applied to different applications in microsystems. In addition to allowing to retrieve the fields inside the devices, it enables to compute more general relations, such as displacement-voltage curves which are the most important characteristics for an electrostatic actuator. Moreover, studying the different applications offers the opportunity to discuss interesting mechanisms such as the pull-in instability often encountered in MEMS devices.

1 Linear FEM Solver

1.1 Theoretical considerations and numerical implementation

The Finite Element Method for linear elasticity allows to retrieve an approximate solution to the following equilibrium equations in the domain Ω_{FEM}:

$$\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} = \boldsymbol{0} \quad , \quad \boldsymbol{\sigma} = \boldsymbol{\sigma}^T \quad \text{and} \quad \boldsymbol{\sigma} = \mathbb{H} : \boldsymbol{\varepsilon}, \tag{1}$$

in which $\boldsymbol{\sigma}$ [N/m^2] denotes the Cauchy stress tensor, ρ [kg/m^3] the density, $\boldsymbol{b}$ [m/s^2] the body forces, $\boldsymbol{\varepsilon}$ [-] the Cauchy strain tensor and $\mathbb{H}$ [N/m^2] the elastic Hooke's tensor. The approximate solution should also satisfy the following boundary conditions on Γ, the boundary of Ω_{FEM}:

$$\boldsymbol{u} = \bar{\boldsymbol{u}}, \quad \text{on } \Gamma_{\bar{u}} \quad \text{and} \quad \boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}}, \quad \text{on } \Gamma_{\bar{t}}, \tag{2}$$

with $\boldsymbol{u}$ [m] the displacement, $\boldsymbol{n}$ [-] the outward unit normal, $\Gamma_{\bar{u}} \cup \Gamma_{\bar{t}} = \Gamma$ and $\Gamma_{\bar{u}} \cap \Gamma_{\bar{t}} = \emptyset$. Last equality implies that some boundary condition must be applied on every boundary of the domain, which corresponds to apply $\bar{\boldsymbol{t}} = \boldsymbol{0}$ on the boundaries where no displacement and no particular surface traction are applied.

A weak formulation of these equations can be derived, involving integrals over the whole domain Ω_{FEM} and its boundary Γ. In the finite element approximation, the domain is divided into several simpler parts called finite elements. In each isoparametric element, the geometry (x, y coordinates) is mapped from a reference element (ξ, η reduced coordinates) as:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{k=1}^{m} N^k(\boldsymbol{\xi}) \boldsymbol{x}^k, \tag{3}$$

in which m denotes the number of nodes of the element and N^k is called a shape function. In one element, the displacement is approximated using the exact same set of shape functions:

$$\boldsymbol{u}^e = \sum_{k=1}^{m} N^k(\boldsymbol{\xi}) \boldsymbol{u}^k, \qquad \text{or, in matrix form:} \quad \boldsymbol{u}^e = \boldsymbol{N} \boldsymbol{d}^e, \tag{4}$$

$$\text{with} \quad (\boldsymbol{d}^e)^T = \begin{bmatrix} u_x^1 & u_y^1 & u_x^2 & .. & u_y^m \end{bmatrix}, \tag{5}$$

with $\boldsymbol{u}^e$ [m] the elemental displacement field and $\boldsymbol{N}$ [-] the shape function matrix. When injecting the approximations in the weak formulation, one linear system of $2n$ equations (for a mesh with n nodes) must be solved:

$$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}, \quad \text{with} \quad \boldsymbol{K} = \int_{\Omega_{\text{FEM}}} \boldsymbol{B}^T \boldsymbol{H} \boldsymbol{B} \, dV \quad \text{and} \quad \boldsymbol{f} = \int_{\Omega_{\text{FEM}}} \rho \boldsymbol{N}^T \boldsymbol{b} \, dV + \int_{\Gamma_{\bar{t}}} \boldsymbol{N}^T \bar{\boldsymbol{t}} \, dS, \tag{6}$$

$$\text{with} \quad \boldsymbol{B} = \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix} \begin{bmatrix} N^1 & 0 & N^2 & .. \\ 0 & N^1 & 0 & .. \end{bmatrix} = \boldsymbol{\partial N} \quad \text{and} \quad \boldsymbol{H} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}, \tag{7}$$

in which $\boldsymbol{K}$ [N/m^2] denotes the stiffness matrix, $\boldsymbol{f}$ [N/m] the nodal forces vector, $\boldsymbol{B}$ [m^{-1}] the strain-displacement matrix, $\boldsymbol{H}$ [N/m^2] the elastic Hooke's tensor obtained when assuming a plane stress configuration using Voigt's notation, E [N/m^2] the Young's modulus of the material and ν [-] its Poisson's ratio.
The main unknowns of the problem are the nodal displacements $\boldsymbol{d}$ [m].

The very first step of the implementation is the mesh generation using the `Gmsh` library and the parameters from the `.geo` file. The second step is the recovery of the material parameters

from the `.geo` file. Then, the structural stiffness matrix $\boldsymbol{K}$ is computed by performing the integral over the whole domain as a sum of integrals over single finite elements. It corresponds to computing the elemental stiffness matrix $\boldsymbol{K}^e$ for each element and then assemble its non-zero elements into the structural stiffness matrix. To exploit the sparsity of $\boldsymbol{K}$, it is declared as a `SparseMatrix` object from the `Eigen` library. During the whole assembly process, the non-zero elements are first stored in a vector of `Triplet` objects from the `Eigen` library, containing the value of the element as well as its two indices. The structural $\boldsymbol{K}$ matrix is eventually generated at once using the complete triplet vector. The elemental $\boldsymbol{K}^e$ matrix (of size $2m \times 2m$, with m the number of nodes of one particular element), is computed using Gauss integration:

$$K_{ij}^e = \int_{-1}^{1}\int_{-1}^{1} B_{ki}(\boldsymbol{\xi}) H_{kl} B_{lj}(\boldsymbol{\xi}) \ \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}\right) \ d\boldsymbol{\xi} \tag{8}$$

$$\approx \sum_{p=1}^{N^{GP}} B_{ki}(\boldsymbol{\xi}^p) H_{kl} B_{lj}(\boldsymbol{\xi}^p) \ \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}^p)\right) \ w_p, \tag{9}$$

with N^{GP} the number of Gauss integration points (automatically fixed by the code, as discussed in Section 1.5), $\boldsymbol{\xi}^p$ the reference coordinates of the Gauss points and w_p the Gaussian weights. The determinants involved in the computation are also called jacobians (jacobian matrix elements defined as $J_{ij} = \partial_{\xi_j} x_i$) and are retrieved using the `Gmsh` library, as well as $\boldsymbol{\xi}^p$ and w_p. As mentioned above, the computation of the elemental $\boldsymbol{B}^e$ involves the derivatives of the shape functions in the geometrical space, which can be computed using the chain rule, for example:

$$\partial_x N^1 = \frac{\partial N^1}{\partial x} = \frac{\partial \boldsymbol{\xi}}{\partial x} \cdot \nabla_{\boldsymbol{\xi}} N^1 \quad \Rightarrow \quad \nabla_{\boldsymbol{x}} N^1 = \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{x}} \cdot \nabla_{\boldsymbol{\xi}} N^1 = \boldsymbol{J}^{-T} \cdot \nabla_{\boldsymbol{\xi}} N^1, \tag{10}$$

in which $\nabla_{\boldsymbol{\xi}} N^1$ can be retrieved once for each element type, while the inverse of the transposed jacobian matrix must be retrieved for each element.
Eventually, the non-zero elements of $\boldsymbol{K}^e$ are assembled in the sparse $\boldsymbol{K}$ matrix, using a mapping between the node tags and the global nodal degrees of freedom: the first node has two degrees of freedom, u_x^1 and u_y^1, corresponding to the indices 0 and 1 of $\boldsymbol{K}$, $\boldsymbol{f}$ and $\boldsymbol{d}$; the second node corresponds to the indices 2 and 3, ...

The volume contribution to the $\boldsymbol{f}$ vector, denoted as $\boldsymbol{f}_V$, is computed in a similar fashion, by first computing the elemental $\boldsymbol{f}_V^e$ vector using Gauss integration:

$$(\boldsymbol{f}_V^e)_i \approx \sum_{p=1}^{N^{GP}} \rho N_{ji}(\boldsymbol{\xi}^p) b_j \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}^p)\right) \ w_p, \tag{11}$$

in which the body forces and the density are assumed constant. Note that using the shape functions ensures *energy consistent nodal loading* ([1]), meaning the external forces are distributed to the nodes in a consistent way regarding the finite element discretization. Afterwards, the global volumic $\boldsymbol{f}_V$ vector is assembled using the same mapping as previously described.

The surface contribution to the $\boldsymbol{f}$ vector, denoted as $\boldsymbol{f}_S$, takes the Neumann boundary conditions into account. It is also computed using Gauss integration, but as the domain of integration corresponds to boundaries, the considered elements are now one-dimensional.

Once again, the elemental $\boldsymbol{f}_S^e$ are first computed using Gauss integration:

$$(\boldsymbol{f}_S^e)_i \approx \sum_{p=1}^{N_{1D}^{GP}} N_{ji}(\boldsymbol{\xi}^p)\bar{t}_j \det\left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\xi}}(\boldsymbol{\xi}^p)\right) w_p, \tag{12}$$

in which a peculiar attention must be paid to the one-dimensional aspect of the integration. After assembling the elemental $\boldsymbol{f}_S^e$ into the global $\boldsymbol{f}_S$, the full right-hand side vector is retrieved as $\boldsymbol{f} = \boldsymbol{f}_V + \boldsymbol{f}_S$.

After the Neumann boundary conditions ($\bar{\boldsymbol{t}}$) have been taken into account, it remains to treat the Dirichlet boundary conditions ($\bar{\boldsymbol{u}}$). It is done by splitting both $\boldsymbol{d}$ and $\boldsymbol{f}$ vectors into two parts as:

$$\begin{bmatrix} \boldsymbol{K}_{RR} & \boldsymbol{K}_{RG} \\ \boldsymbol{K}_{GR} & \boldsymbol{K}_{RR} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_R \\ \bar{\boldsymbol{d}}_G \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{f}}_R \\ \boldsymbol{f}_G \end{bmatrix}, \tag{13}$$

in which $\bar{\boldsymbol{d}}_G$ and $\bar{\boldsymbol{f}}_G$ are given and known through boundary conditions. Hence, the linear system to solve can be reduced to:

$$\boldsymbol{K}_{RR}\boldsymbol{d}_R = \bar{\boldsymbol{f}}_R - \boldsymbol{K}_{RG}\bar{\boldsymbol{d}}_G, \tag{14}$$

in which the only unknown is $\boldsymbol{d}_R$ (notations from [1]). This allows to reduce the dimensions of the linear system and to treat simultaneously both homogeneous and inhomogeneous Dirichlet boundary conditions (a homogeneous boundary conditions does not induce any correction term).

Eventually, the reduced sparse linear system is solved using a `SimplicialLDLT` object from the `Eigen` library, which implements an efficient direct solving algorithm for sparse, symmetric and positive definite matrices ($\boldsymbol{K}_{RR}$ is positive definite if the Dirichlet boundary conditions prevent from any rigid body mode). According to the Eigen documentation, it is *"recommended for very sparse and not too large problems (e.g., 2D Poisson eq.)"*[1]. It is preferred over iterative solving algorithms, which are less predictable, can diverge in more situations than direct solvers and require more computational time to reach the same accuracy. The performance of the `SimplicialLDLT` solver is compared to other solvers in Section 4.5.

Afterwards, the complete nodal displacement vector $\boldsymbol{d}$ is reconstructed using the prescribed boundary conditions and the full nodal forces vector $\boldsymbol{f}$ is retrieved by computing $\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f}$.

The reaction forces $\boldsymbol{R}$ can be retrieved on boundaries where some component of the displacement has been fixed, by summing the elements in $\boldsymbol{f}_G$ corresponding to the nodal displacements which have been imposed by the Dirichlet boundary conditions. Note that the reaction force along one particular axis is computed by summing only the elements corresponding to nodal displacement along the particular axis.

The nodal strain values and stress values can be retrieved in each finite element from the nodal displacements $\boldsymbol{d}$:

$$\begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \boldsymbol{B}\boldsymbol{d}^e \qquad \text{and} \qquad \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{bmatrix}, \tag{15}$$

in which Voigt's notation is used. As the considered equations are linear, the strains and stresses can be computed directly at the nodes of each element (if the equations were not

[1] Cited from the Eigen documentation regarding solving sparse linear systems, consulted on May 13, 2022.

linear, they should have been computed at the Gauss points). Note that the obtained fields are not necessarily continuous across element boundaries. From the plane stress assumption, the principal strain in the plane direction is retrieved as $\varepsilon_{zz} = -\frac{\nu}{E}(\sigma_{xx} + \sigma_{yy})$ and the equivalent von Mises stress as $\sigma_{VM} = \sqrt{\sigma_{xx}^2 + \sigma_{yy}^2 - \sigma_{xx}\sigma_{yy} + 3\sigma_{xy}^2}$.

One can also evaluate the so-called total potential energy TPE [J/m], introducing the internal strain energy U [J/m] and the work done by external forces P [J/m], defined as

$$TPE = U - P, \quad \text{with} \quad U = \int_{\Omega_{\text{FEM}}} \frac{1}{2}\boldsymbol{\sigma} : \boldsymbol{\varepsilon}\, dV \quad \text{and} \quad P = \int_{\Omega_{\text{FEM}}} \boldsymbol{b}\cdot\boldsymbol{u}\, dV + \int_{\Gamma_t} \bar{\boldsymbol{t}}\cdot\boldsymbol{u}\, dS. \tag{16}$$

The internal strain energy U is computed as a sum of integrals over single finite elements and using Gauss integration, similarly to the computation of the stiffness matrix. The work done by external forces P can be computed easily by multiplying the final nodal displacements $\boldsymbol{d}$ and the final nodal forces $\boldsymbol{f}$, as the nodal forces have been computed in an energy consistent way. Moreover, for the present linear theory of elasticity, Clapeyron's theorem [1] states that the internal strain energy is equal to the half of the work done by external forces $U = P/2$. Hence, the TPE is given by

$$TPE = U - P = -U = -\frac{P}{2}. \tag{17}$$

Eventually, the different results are saved and displayed on the mesh.

1.2 Validation in simple tension configuration

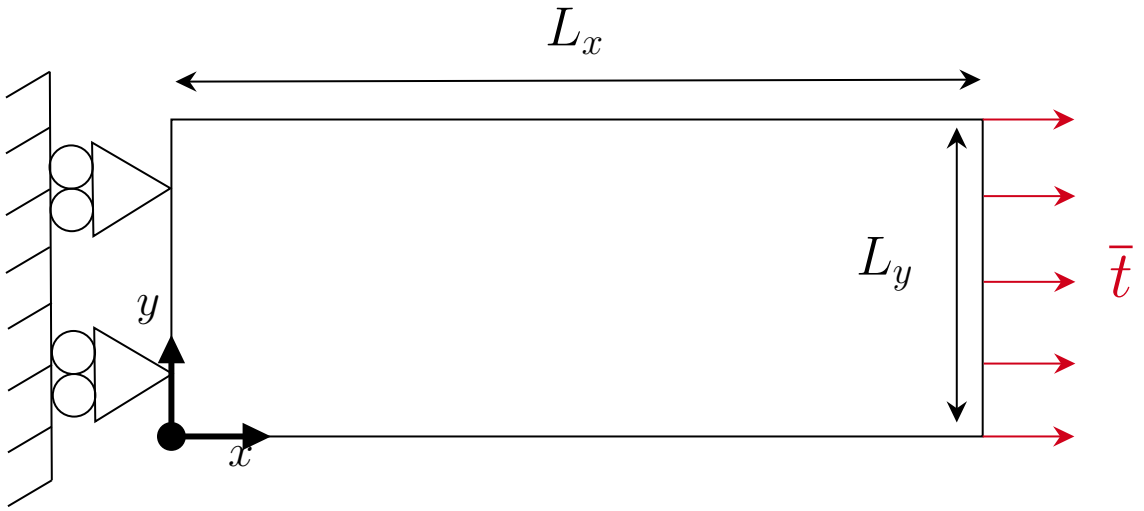


Figure 1: Geometry and boundary conditions associated to the simple tension configuration. No horizontal displacement is allowed on the left edge, while the vertical displacement is prescribed to $\bar{u}_y = 2$ [m] at the point at the bottom left. The surface tension applied on the right edge is $\bar{t} = 21\ 10^3$ [Pa], while the Young's modulus of the bar is $E = 210\ 10^3$ [Pa] and its Poisson's ratio is $\nu = 0.3$ [-]. The length of the bar is $L_x = 5$ [m] and its height $L_y = 2$ [m].

The first step of the validation involves a simple tension configuration, as described in Figure 1. It is implemented in the `simple_tension.geo` file. The analytical solution corresponds to a uniform normal stress $\sigma_{xx} = \bar{t} = 21\ 10^3$ [Pa] in the whole bar, and $\sigma_{yy} = \sigma_{xy} = 0$ [Pa]. The corresponding axial strain $\varepsilon_{xx} = \bar{t}/E = 0.1$ [-] is also uniform,

while $\varepsilon_{yy} = \varepsilon_{zz} = -\nu\varepsilon_{xx} = -0.03$ [-]. The displacement fields should therefore evolve linearly along the x-axis. Particularly, $u_x(x = L_x) = 0.5$ [m]. The internal strain energy should be equal to $U = L_x\, L_y\, \sigma_{xx}\, \varepsilon_{xx}/2 = 10500$ [J/m] and the work done by external forces should be given by $P = L_x\, u_x(x = L_x)\, \bar{t} = 21000$ [J/m].
Some of the numerical results are shown in Figure 2. They are similar to the analytical solution. The horizontal reaction force at the left edge is $R_x = -42\ 10^3$ [N/m] $= -\bar{t}L_y$. The vertical reaction force at the fixed node is $R_y = 1.95\ 10^{-9}$ [N/m] and is negligible. The numerical internal strain energy is exactly $U = 10500$ [J/m] and the work done by external forces $P = 21000$ [J/m].

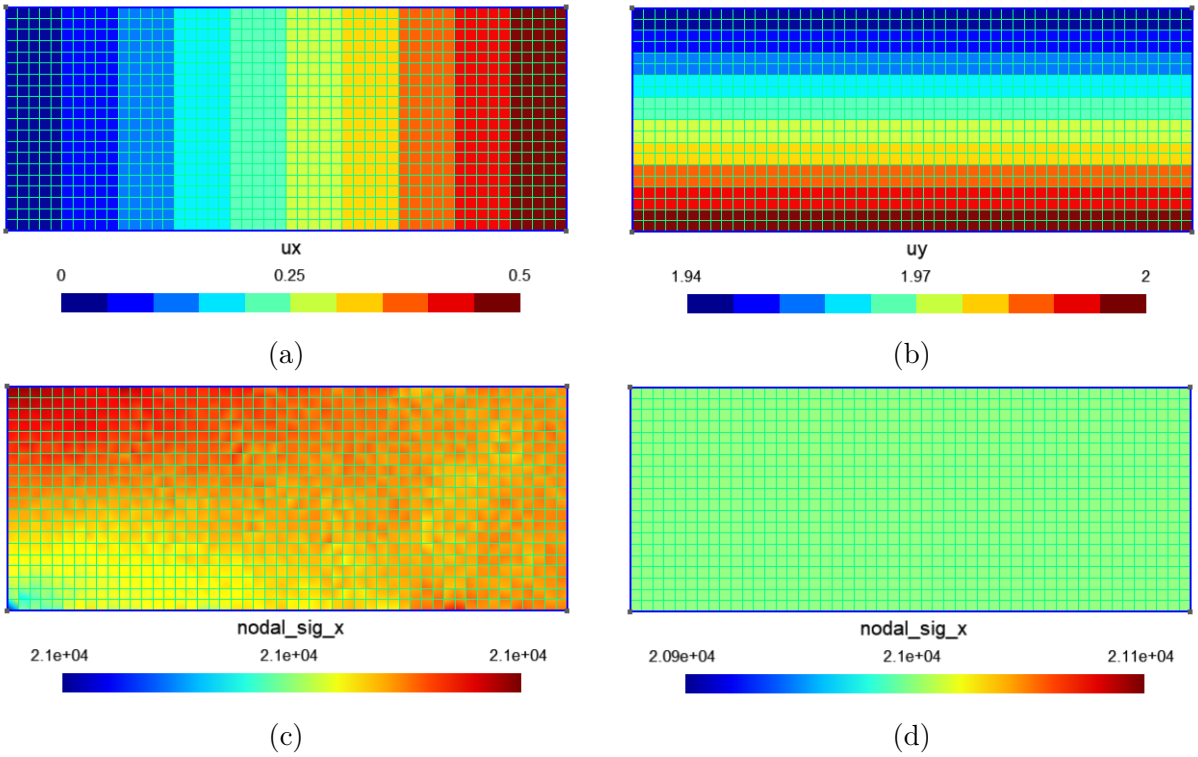


Figure 2: (a) Horizontal displacement field u_x [m], vertical displacement field u_y [m] and (c) normal stress field σ_{xx} [Pa] obtained with the configuration of Figure 1, using a mesh of $n_x = 50$ [-] elements along the x-axis and $n_y = 20$ [-] along the y-axis, resulting in square elements of side 0.1 [m]. In (d), the color bar range has been modified to $[20.9 - 21.1]\ 10^3$ [Pa] to highlight that the variations are negligible.

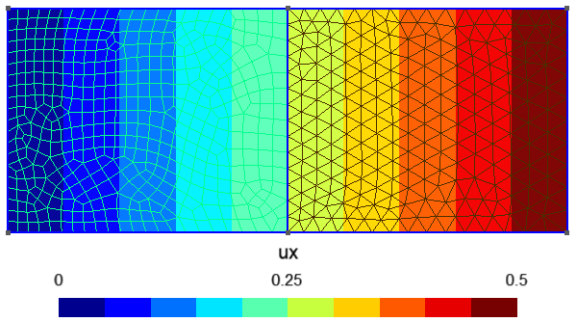


Figure 3: Horizontal displacement field u_x [m] obtained with an hybrid mesh generated with the `complex_validation.geo` file and elements of the second order.

The code has been validated for both triangular and quadrangular elements. Moreover, it works with many different integration rules and it also works with elements of higher order. As can be seen in Figure 3, the obtained results are similar to the ones showed previously.

1.3 Validation: clamped beam with uniform vertical charge

The second validation configuration is more complex and is described in Figure 4. It is implemented in the `uniform_charge.geo` file. The focus is set on the vertical deflection at the right edge and on the vertical reaction force at the left edge. As $L_x \gg L_y$, the Euler-Bernoulli beam theory can be used. In this configuration, the vertical deflection at the end of the beam should be $v_R = \bar{p}L_x^4/8EI_z = 3\bar{p}L_x^4/2EL_y^3 = 0.0714$ [m]. The vertical reaction force is simply $R_y = \bar{p}L_x = 10$ [N/m].

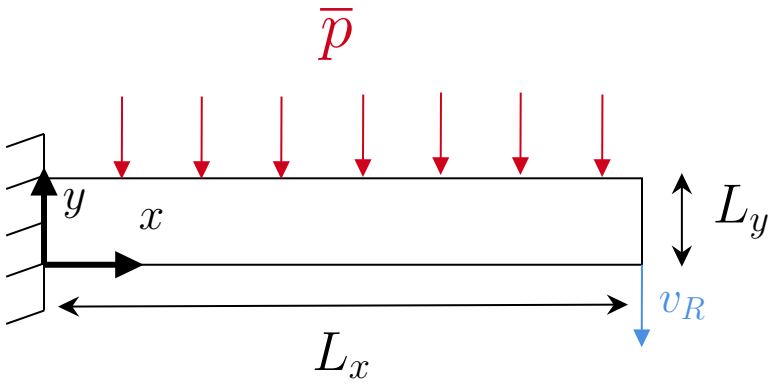


Figure 4: Geometry and boundary conditions associated to a clamped beam with uniform vertical charge. No horizontal and vertical displacement is allowed on the left edge. The surface tension applied on the top edge is $\bar{p} = 1$ [Pa], while the Young's modulus of the bar is $E = 210\ 10^3$ [Pa] and its Poisson's ratio is $\nu = 0.3$ [-]. The length of the bar is $L_x = 10$ [m] and its height $L_y = 1$ [m]. Its bending moment (per unit thickness) around the perpendicular axis is $I_z = L_y^3/12 = 0.083$ [m^3].

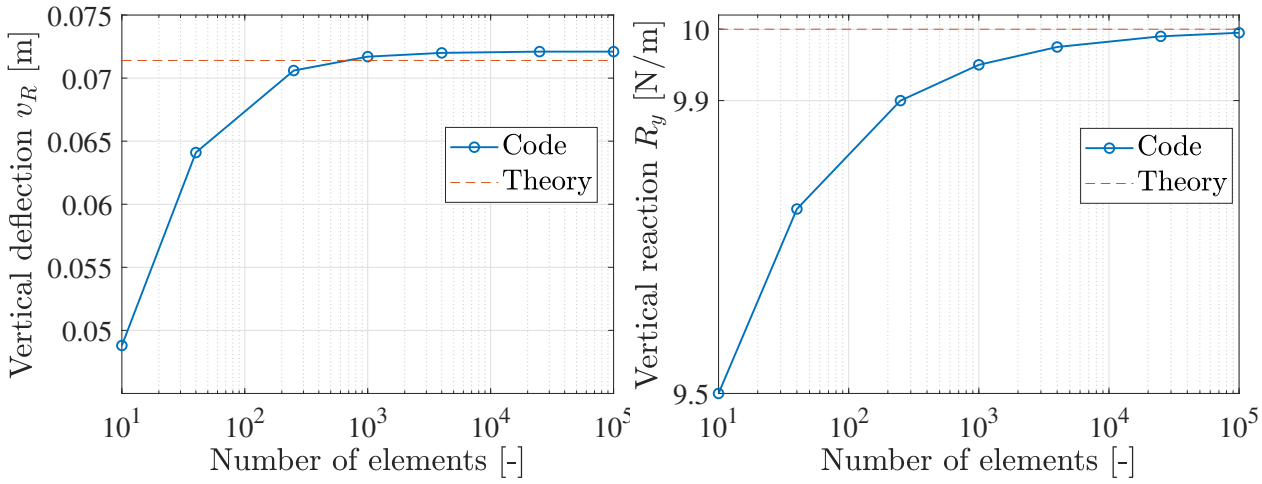


Figure 5: Convergence of (a) the vertical deflection v_R [m] and (b) the vertical reaction force R_y [N/m] for the clamped beam geometry, using first order square elements.

The numerical results are obtained with first order square elements of decreasing size (or increasing number). The convergence of both values of v_R and R_y as the number of elements increases is shown in Figure. 5.

As can be observed, the numerical results converge towards the theoretical values, which shows the accuracy of the solver. Note that the vertical deflection converges towards $v_R = 0.0721$ [m], which is slightly greater than the theoretical value. It could be due to the restrictive assumptions made in the theoretical development that are not fully respected.

1.4 Comparison of different finite elements

In this section, general results of the FEM code in the clamped beam configuration are studied as a function of four different element types: quadrangular elements of first and second order, triangular elements of first order and second order, denoted respectively $Q4$ and $Q9$, $T3$ and $T6$.

1.4.1 CPU time

First, the evolution of the CPU time as a function of the number of elements and the number of nodes, represented in Figure 6, is analyzed. For the clarification, all tests in this report were performed on a MacBook Pro with a 2.5 GHz Intel Core i7 quad-core processor and 16 GB DDR3 ram.

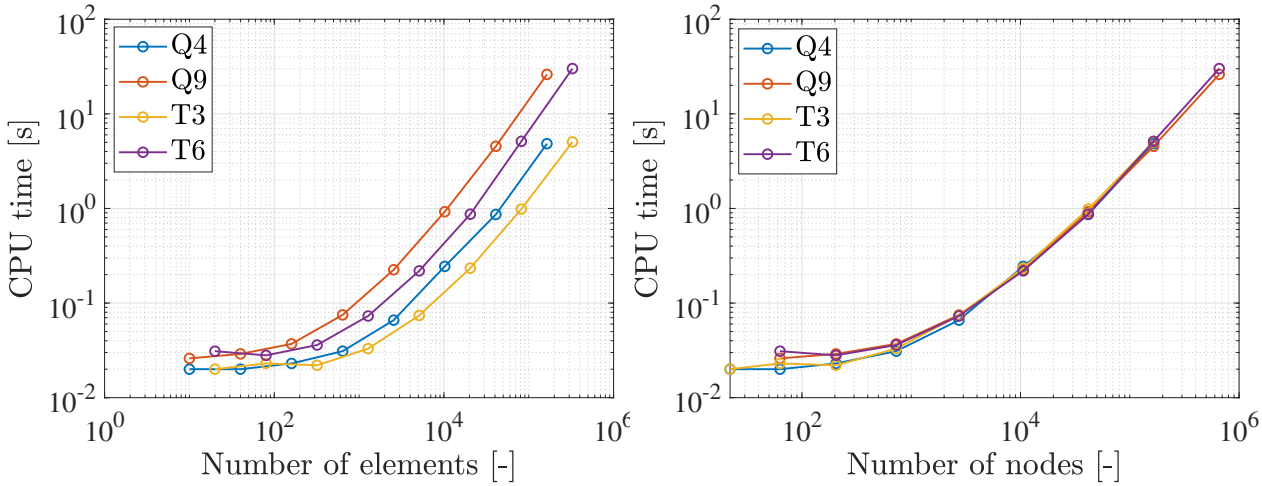


Figure 6: CPU time [s] as a function of (a) the number of elements and (b) the number of nodes, for the clamped beam geometry described in Figure 4. Measurements performed using 8 virtual threads.

As can be observed, for a given number of finite elements, second order elements ($T6$ and $Q9$) take the most time to perform the computation. Moreover, for a given order of finite elements, the quadrangular elements are the more time-consuming elements. This can be explained by the fact that the size of the numerical problem is directly proportional to the number of nodes in the domain (in 2D, the number of degrees of freedom is twice the number of nodes) and not to the number of elements. Indeed, as the number of nodes increases, the CPU time is similar for all four types of elements.
In practice, as the size of the problem increases, the total CPU time is more and more

dominated by the time taken to solve the linear system numerically. This phenomenon is discussed in Section 4.5 in the context of the non-linear FEM solver dealing with large displacements. Despite the differences in the algorithm, similar conclusions can be drawn in the present linear case.

1.4.2 Vertical deflection and vertical reaction force

Moreover, one can also compare the exact same quantities than the one discussed in Section 1.3, the vertical deflection v_R and the vertical reaction force R_y. The evolution of both quantities as a function of the total number of nodes for the different element types is represented in Figure 7. As can be observed for the vertical deflection, the $T3$ elements behave poorly compared to the other element types. Also, the second order elements converge much more rapidly than the first order elements. However, the $Q4$ and $Q9$ converge respectively faster than the $T3$ and $T6$ elements. A similar behaviour is also observed when focusing on the convergence of the vertical reaction force. Note that in the present case, the curves for the $T3$ and $T6$ elements are indistinguishable, as well as for the $Q4$ and $Q9$ elements. In such a bending configuration, quadrangular elements are preferred over triangular elements.

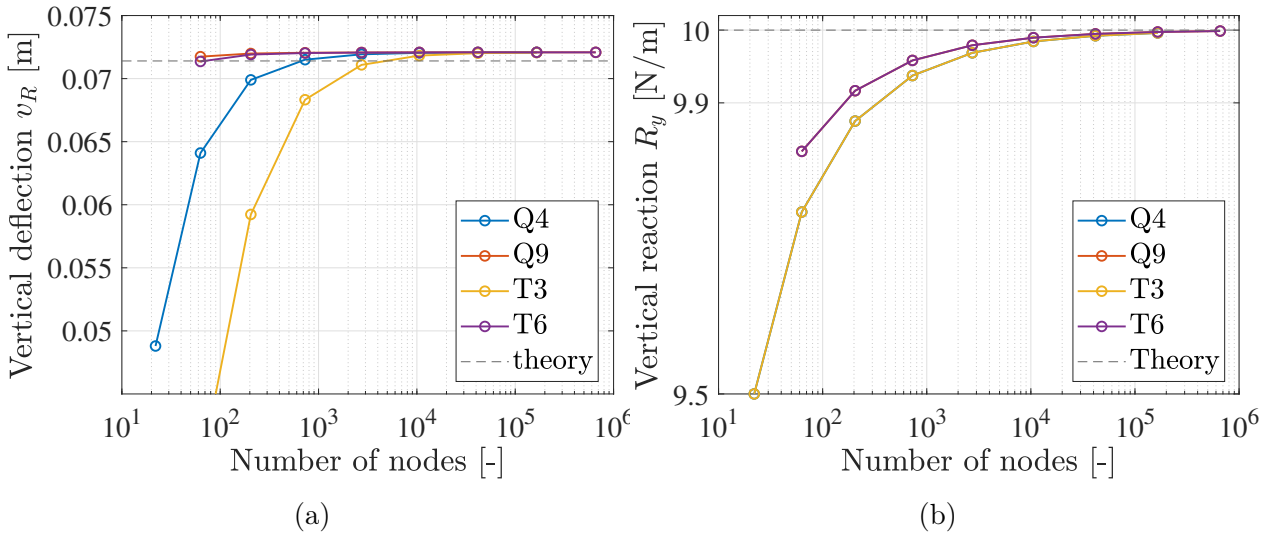


Figure 7: Convergence of (a) the vertical deflection v_R [m] and (b) the vertical reaction force R_y [N/m] for the clamped beam geometry as a function of the number of nodes for the four different element types.

1.4.3 Maximal von Mises stress

When comparing the behaviour of the different element types regarding the maximal von Mises stress σ_{VM} in the bar, as represented in Figure 8, one can first observe that the stress field does not seem to have converged yet, independently from the type of the finite element. Most often, a precise elemental-nodal stress computation requires a much finer mesh than a precise displacement computation. This can be explained by the fact that the main unknowns of the finite element resolution are the nodal displacements, and not the stresses. Hence, despite a *strong* knowledge on displacement, the *weak* knowledge on the stress field is the price to pay for the finite element approximation, as explained in [1].

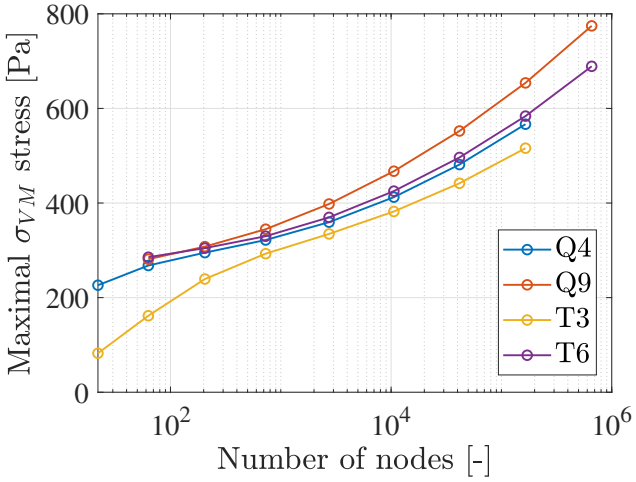


Figure 8: Evolution of the maximal von Mises stress σ_{VM} [Pa] for the clamped beam geometry as a function of the number of nodes for the four different element types.

1.4.4 Total potential energy

Finally, the convergence of the total potential energy TPE, also called global convergence, can be studied. It is represented in Figure 9(a). Once again, the convergence is much faster for second order elements. As expected, the convergence is monotonic and the TPE is minimal (its absolute value is maximal) at equilibrium when the mesh is refined.

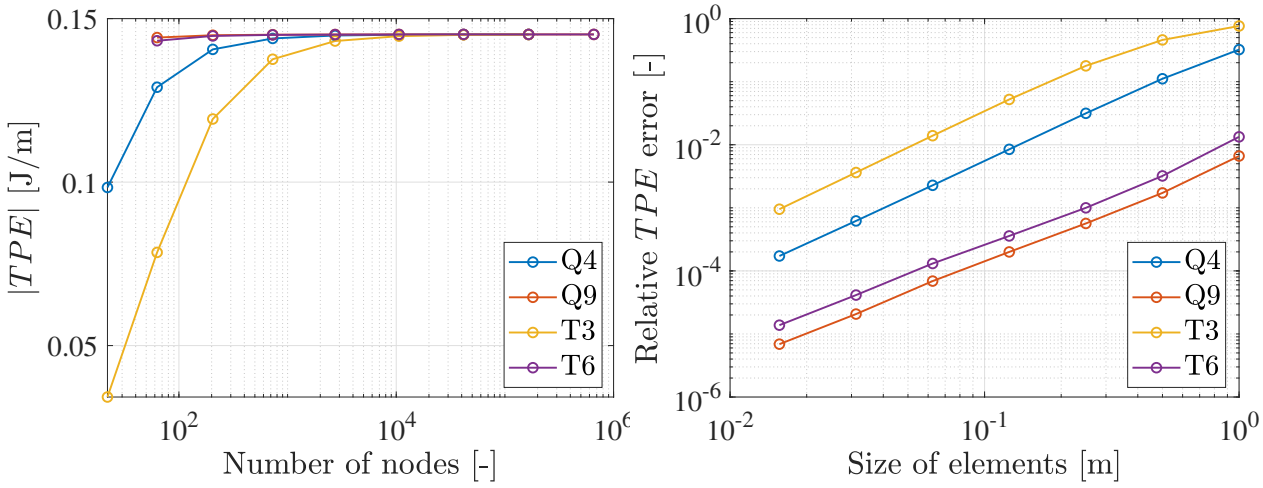


Figure 9: (a) Convergence of the absolute value of the total potential energy $|TPE|$ [J/m] as a function of the number of nodes. (b) Evolution of the relative error on the total potential energy as a function of the size of the elements. The results are displayed for the four different element types.

The relative error on the TPE can be computed as

$$\left| \frac{TPE_{\text{approx}} - TPE_{\text{exact}}}{TPE_{\text{exact}}} \right|, \tag{18}$$

with $TPE_{\text{exact}} = -0.145165$ [J/m] the converged value of the total potential energy. As can be observed in Figure 9(b), the relative error decreases when the size of the finite elements decreases (the graph being in logarithmic axis). Moreover, the convergence of the relative error seems to be of order two independently from the element type, as the relative error is multiplied by four when the size of the elements is doubled.

1.5 Impact of the Gauss integration rule

In this section, the impact of the order of the Gauss integration is discussed in the case of quadrangular elements, as they are more accurate in the case of finite element computation. The implementation of the finite element method requires numerical integration of different quantities, as described in Section 1.1. Most importantly, the elemental stiffness matrices are computed using Gauss integration (Equation 9). In particular, a `CompositeGauss` method is used, which is an extension of the one dimensional Gauss-Legendre integration method. It requires the same number n_{GP} of Gauss points along both x and y directions, for a total number of Gauss points $N_{\text{GP}} = n_{\text{GP}}^2$.

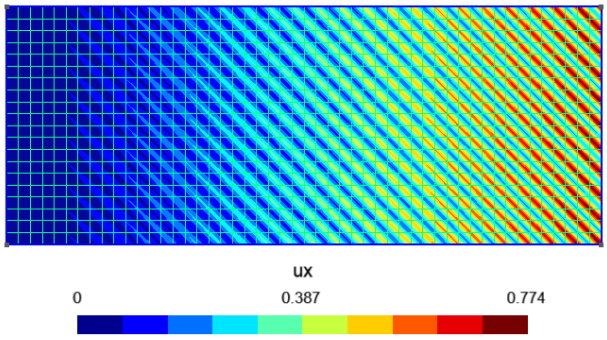
In one dimension, the Gauss-Legendre method with n_{GP} Gauss points integrates exactly polynomials of order $2n_{\text{GP}}$-1 and below. For integrating same order polynomials in two dimensions, n_{GP} Gauss points are required in each direction.

For rectangular elements (specific case of quadrangular elements), it has been shown in [1] that the stiffness matrix (Equation 9) is exactly integrated when using $N_{\text{GP}} = 2^2 = 4$ Gauss points for $Q4$ elements or $N_{\text{GP}} = 3^2 = 9$ Gauss points for $Q9$ elements. They correspond respectively to the `CompositeGauss2` and `CompositeGauss4` methods implemented in `gmsh`. Different integration methods are tested with different element orders in the simple tension configuration described in Figure 1. The results are gathered in Figure 10. As can be observed, exact integration is required, meaning first order elements need the `CompositeGauss2` method for performing the integration, while second order elements need the `CompositeGauss4` method.

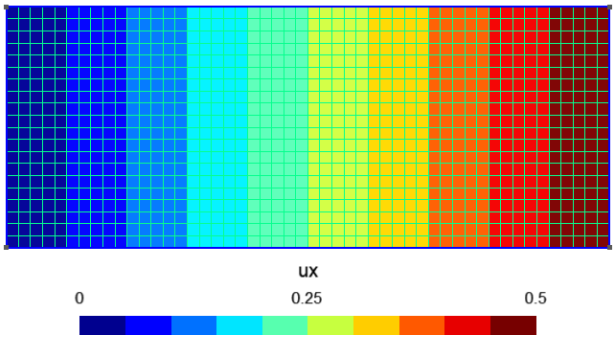
When using less Gauss points than required (and performing reduced integration), one can observe spurious modes that do not represent the physics and the results are less accurate. In this case, the elemental stiffness matrix is said to be rank deficient. Physically, the spurious modes, also called hourglass modes, are local nodal displacements that do not induce any change in total potential energy, as they result from the rank deficiency from the elemental stiffness matrix. Hence, the spurious modes spread in the whole mesh and reduced integration must be avoided.

The required integration method is automatically adapted in the code depending on the element order, when computing the elemental stiffness matrices.

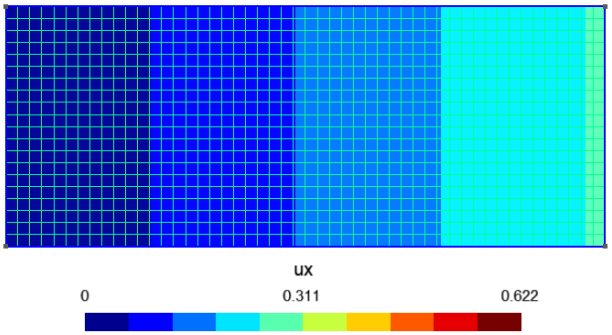
In the case of non rectangular elements (for which the opposite sides are not parallel), the numerical integration is not exact and is called fulled integration. Similarly, $N_{\text{GP}} = 4$ Gauss points are required for $Q4$ elements and $N_{\text{GP}} = 9$ Gauss points are required for $Q9$ elements in order to ensure numerical stability.



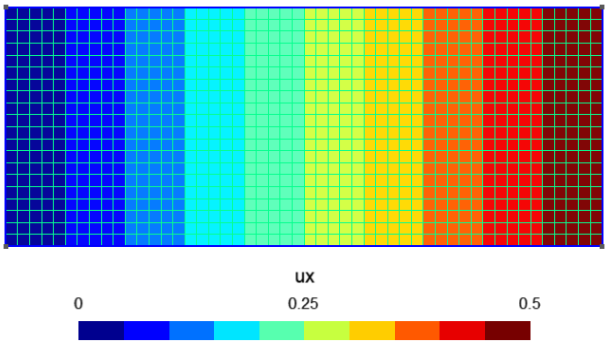
(a) First order elements, $N_{\mathrm{GP}} = 1$.



(b) First order elements, $N_{\mathrm{GP}} = 4$.



(c) Second order elements, $N_{\mathrm{GP}} = 4$.



(d) Second order elements, $N_{\mathrm{GP}} = 9$.

Figure 10: Horizontal displacement field u_x computed in the simple tension configuration described in Figure 1 for different element orders and using different number of Gauss integration points N_{GP}.

2 BEM Solver

2.1 Theoretical approach

This first part of the theory and numerical implementation of the Boundary Element Method is mainly based on the book by M. Katsikadelis [3]. The aim of this theoretical section is to introduce the boundary element method for solving the electrostatics problem in two dimensions. The latter is governed by the Laplace equation for the electric potential ϕ [V]:

$$\Delta\phi = 0, \text{ in } \Omega_{\text{BEM}}, \tag{19}$$

which is valid in every domain Ω_{BEM} in which there is no free charge density ρ [C/m^3]. Indeed, when $\rho = 0$ [C/m^3], Gauss's law for the electric field $\boldsymbol{E}$ [V/m] reduces to

$$\nabla \cdot \boldsymbol{E} = 0. \tag{20}$$

As the electric potential is defined such that

$$\boldsymbol{E} = -\nabla\phi, \tag{21}$$

Laplace equation for the electric potential is simply retrieved as:

$$-\nabla \cdot \boldsymbol{E} = \nabla \cdot (\nabla\phi) = \Delta\phi = 0. \tag{22}$$

The Dirichlet and Neumann boundary conditions on the boundary are given by

$$\phi = \bar{\phi}, \text{ on } \Gamma_D, \qquad \text{and} \qquad \frac{\partial\phi}{\partial\mathbf{n}} = \bar{\phi}_n, \text{ on } \Gamma_N. \tag{23}$$

where $\Gamma = \Gamma_D \cup \Gamma_N$ represents the curve that delimits the boundary of the surface. With the property that $\Gamma = \Gamma_D \cap \Gamma_N = \emptyset$. The Γ_D and Γ_N curves are those where the Dirichlet and Neumann conditions apply respectively. The electric potential ϕ is the unknown of the problem and is related to Dirichlet boundary conditions. If $\mathbf{n}$ represents the exterior normal to the Ω boundary, one can write $\frac{\partial\phi}{\partial\mathbf{n}} = \nabla\phi \cdot \mathbf{n} = -\boldsymbol{E} \cdot \boldsymbol{n}$, Neumann boundary conditions are thus associated to the electrical field.

A fundamental solution of the Laplace Equation 19 is given by considering a Dirac impulse in the right-hand member and ignoring the boundary conditions:

$$\Delta\varphi = \delta(\mathbf{Q} - \mathbf{P}) \;\Leftrightarrow\; \varphi(\mathbf{Q}, \mathbf{P}) = \frac{1}{2\pi}\ln(r), \tag{24}$$

with r [m] the distance between the points located in $\boldsymbol{P}$ and $\boldsymbol{Q}$.

As a reminder, Green's second identity is given by:

$$\int_\Omega (\varphi\Delta\phi - \phi\Delta\varphi)\, d\Omega = \int_\Gamma \left(\varphi\frac{\partial\phi}{\partial\mathbf{n}} - \phi\frac{\partial\varphi}{\partial\mathbf{n}}\right) ds. \tag{25}$$

Applying this expression to our problem, one obtains the representation formula:

$$\phi(\mathbf{P}) = -\int_\Gamma \left(\varphi(\mathbf{P}, \mathbf{q})\frac{\partial\phi(\mathbf{q})}{\partial\mathbf{n_q}} - \phi(\mathbf{q})\frac{\partial\varphi(\mathbf{P}, \mathbf{q})}{\partial\mathbf{n_q}}\right) ds_{\mathbf{q}}. \tag{26}$$

The subscripts $\mathbf{q}$ denotes the points that vary during integration or differentiation. The capital letters $(\mathbf{P}(x,y), \mathbf{Q}(\xi,\eta))$ represent points inside the domain while the lower case letters $(\mathbf{p}(x,y), \mathbf{q}(\xi,\eta))$ represent a point on the boundary.
This first equation allows us to calculate the value of ϕ inside the domain provided $\phi(\mathbf{q})$ and $\frac{\partial\phi(\mathbf{q})}{\partial\mathbf{n}_q}$ are known. An equation is missing to satisfy this last condition. Once again, using Green's identity helps to find a representation formula at the boundary. Considering that the boundary is smooth, it comes that:

$$\frac{1}{2}\phi(\mathbf{p}) = -\int_\Gamma \left(\varphi(\mathbf{p},\mathbf{q})\frac{\partial\phi(\mathbf{q})}{\partial\mathbf{n_q}} - \phi(\mathbf{q})\frac{\partial\varphi(\mathbf{p},\mathbf{q})}{\partial\mathbf{n_q}}\right) ds_\mathbf{q}. \tag{27}$$

Equation 27 will allow us to calculate $\phi(\mathbf{q})$ and $\frac{\partial\phi(\mathbf{q})}{\partial\mathbf{n_q}}$ at the boundary (where it is unknown) and then find the value of ϕ over the entire domain from Equation 26. The BEM will consist in discretising the boundary in finite elements. The chosen approach is to use linear elements with a constant ϕ solution and derivative $\frac{\partial\phi}{\partial\mathbf{n}}$ over each element. The first step is to discretize the Equation 27 as follow:

$$\frac{1}{2}\phi_i = -\sum_{j=1}^{N}\int_{\Gamma_j}\varphi(\mathbf{p}_i,\mathbf{q})\underbrace{\boxed{\frac{\partial\phi(\mathbf{q})}{\partial\mathbf{n_q}}}}_{\phi_n^j} ds_\mathbf{q} + \sum_{j=1}^{N}\int_{\Gamma_j}\underbrace{\boxed{\phi(\mathbf{q})}}_{\phi_j}\frac{\partial\varphi(\mathbf{p}_i,\mathbf{q})}{\partial\mathbf{n_q}} ds_\mathbf{q}. \tag{28}$$

Defining the matrices H and G as

$$H_{ij} = \int_{\Gamma_j}\frac{\partial\varphi(\mathbf{p}_i,\mathbf{q})}{\partial\mathbf{n_q}} ds_\mathbf{q} - \frac{1}{2}\delta_{ij} \qquad \text{and} \qquad G_{ij} = \int_{\Gamma_j}\varphi(\mathbf{p}_i,\mathbf{q}) ds_\mathbf{q}, \tag{29}$$

the system of equations can be rewritten

$$\sum_{j=1}^{N} H_{ij}\phi^j = \sum_{j=1}^{N} G_{ij}\phi_n^j \tag{30}$$

2.2 Implementation of the representation formula on the boundary

From a numerical point of view, the elements are first stored in a vector of a particular `element structure` storing every useful information concerning a given element, that will be needed for the further computations (and for debugging possibly). This structure is composed of

- a tag;
- a type of element (Dirichlet or Neumann): Thanks to the `ONELAB` database, it is possible to extract the type of boundary condition and its value assigned to each physical group in the `.geo` file. Each element being part of a given physical group then inherits from its properties;
- an array for storing the potential and its directional derivative: One of these quantity is given by the boundary condition (as described above) and the other has to be computed. One should pay attention to the fact that, depending on the type of boundary condition imposed on the element, these values will not be stored at the same indices after the

same operation (the boundary condition should be stored at index 0 for a Dirichlet boundary condition and the computed value at index 1 and inversely for a Neumann boundary condition);

- an array for storing the tags of the extremities nodes composing the element: When looping over the elements, the tags of the nodes of an element are stored in this array. It is important to store it in the same order as they are given by the `.geo` file. Otherwise, it would result in a misorientation of the normal of the elements;
- the tag of the mid element node. In fact, for higher order elements, there is a node at the middle of each boundary element. If the element does not possess such a node, this tag is set to 0. In fact, since there is no tag 0, this correspond to an *error / default* value;
- some other geometric parameters such as the length of the element and the coordinates of the nodes.

A particularity of this vector is that it should be sorted, so that all the Dirichlet elements are before all the Neumann elements. In order to do this, the simplest thing is to append directly the elements at the front or at the back of the vector depending on their type. Also, counting the number of Dirichlet elements permits to delimit the space of Dirichlet/Neumann elements in this vector.

The following of the implementation will mainly consist in computing the H_{ij} and G_{ij} components and to solve the system from Equation 30 for the unknowns. The way of computing such components is to make a double loop over the elements of the vector.

In order to better visualise the operations of the BEM, the main notations are introduced in Figure 11.

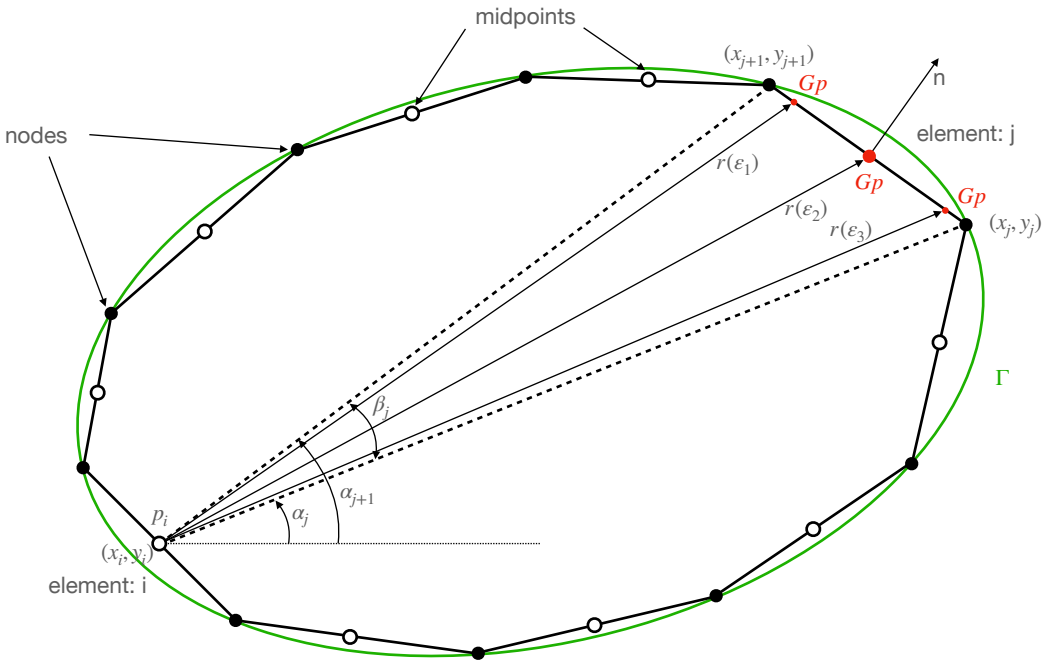


Figure 11: Diagram and highlighting of the main BEM variables.

The individual G_{ij} components can be calculated analytically on the diagonal or using a Gaussian integration for the non-diagonal components:

$$G_{ij} = \frac{l_j}{4\pi} \sum_{k=1}^{N^{GP}} \ln(r(\varepsilon_k)) w_k \qquad\qquad G_{ii} = \frac{l_i}{2\pi} \left(\ln \frac{l_i}{2} - 1 \right) \tag{31}$$

where l represent the length of an element and N^{GP} the number of Gauss points considered. They are located at the positions ε_k (in local coordinates) with the associated weights w_k. The expression of the global coordinates as a function of the position of the nodes is:

$$\begin{cases} x(\varepsilon) &= \dfrac{x_{j+1} + x_j}{2} + \dfrac{x_{j+1} - x_j}{2}\varepsilon, \\ y(\varepsilon) &= \dfrac{y_{j+1} + y_j}{2} + \dfrac{y_{j+1} - y_j}{2}\varepsilon. \end{cases} \tag{32}$$

The length of the element and the position vector relative to the Gauss points can be written as:

$$l_j = \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2}, \qquad r(\varepsilon_k) = \sqrt{(x(\varepsilon_k) - x_i)^2 + (y(\varepsilon_k) - y_i)^2} \tag{33}$$

The individual H_{ij} components can be calculated fully analytically as:

$$H_{ij} = \frac{\alpha_{j+1} - \alpha_j}{2\pi} \qquad\qquad H_{ii} = -\frac{1}{2} \tag{34}$$

where α_j and α_{j+1} represent the angles joining the midpoint of element i to the nodes of element j as described in Figure 11. They can be calculated using the relations

$$\tan \alpha_{j+1} = \frac{y_{j+1} - y_i}{x_{j+1} - x_i}, \qquad\qquad \tan \alpha_j = \frac{y_j - y_i}{x_j - x_i}. \tag{35}$$

It is important to note that these formula are derived for positive oriented curves, so that the domain is always on the left of the curve. In the case where holes are present in the domain, one should pay attention to the fact that in order to have a positive oriented curve on the boundary off a hole, the curves should be oriented as shown in Figure 12.

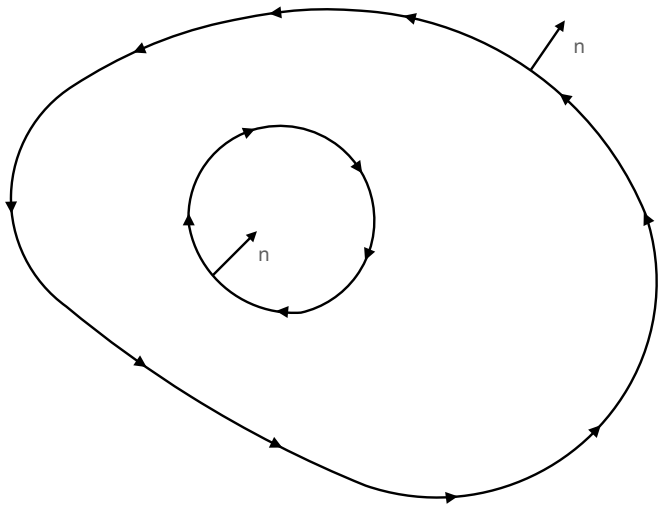


Figure 12: Diagram of the orientation of the curves for a domain containing a hole.

2.2.1 Angles computation

It is possible to continue the algorithm along this path. However, there is a simpler and more robust method for calculating angles. Indeed, it is important to stress that only the difference of the angles α_{j+1} and α_j counts for evaluating the element of the matrix H_{ij}. The real variable of interest is therefore the angle formed between the two vectors starting at p_i and ending at each of the two nodes constituting the j element. This angle is called β_j as shown in Figure 13. It can be evaluated simply by considering the relation of the scalar product between two vectors. However, the order of the vectors is important and will depend on the geometry of the problem. For example, a concave geometry could lead to the inversion of the order of the nodes. To ensure the correct orientation, a vector product is performed. If this is positive, the order of the vectors is correct. However, if it is negative, the order is reversed and the code will provide the opposite of the calculated angle.

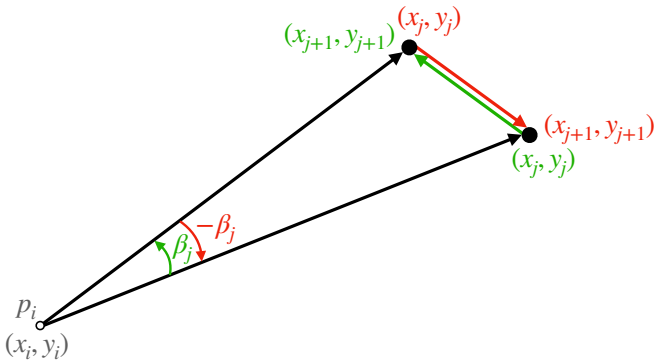


Figure 13: Schematic of the angles difference and orientation.

2.2.2 Building the linear equation system

The next step is to build and to compute a linear system from the $\boldsymbol{H}$ and $\boldsymbol{G}$ matrices.

As an example, let us consider a surface delimited by a boundary consisting of four elements where there are two Dirichlet conditions and two Neumann conditions imposed along the curve. This example is illustrated below in Figure 14.

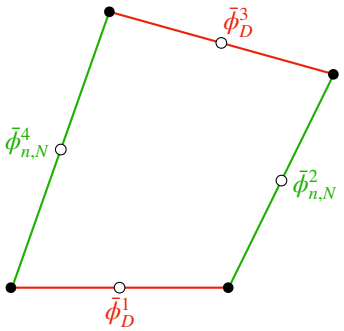


Figure 14: Simple example of a boundary consisting of four elements where two Neumann and Dirichlet conditions are imposed on opposite sides.

In this example, the system can be written in the matrix form as:

$$\begin{bmatrix} H_{11} & H_{12} & H_{13} & H_{14} \\ H_{21} & H_{22} & H_{23} & H_{24} \\ H_{31} & H_{32} & H_{33} & H_{34} \\ H_{41} & H_{42} & H_{43} & H_{44} \end{bmatrix} \begin{bmatrix} \bar{\phi}_D^1 \\ \bar{\phi}_D^3 \\ \phi_N^2 \\ \phi_N^4 \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ G_{31} & G_{32} & G_{33} & G_{34} \\ G_{41} & G_{42} & G_{43} & G_{44} \end{bmatrix} \begin{bmatrix} \phi_{n,D}^1 \\ \phi_{n,D}^3 \\ \bar{\phi}_{n,N}^2 \\ \bar{\phi}_{n,N}^4 \end{bmatrix} \tag{36}$$

The imposed Neumann and Dirichlet conditions on the system are both on the left and on the right of the equality. These are assumed to be known values of the problem. The unknowns that remain should thus be moved to one side in order to construct a linear system of equations encoded in matrix form that can be solved. The system writes as

$$\underbrace{\begin{bmatrix} G_{11} & G_{13} & -H_{12} & -H_{14} \\ G_{21} & G_{23} & -H_{22} & -H_{24} \\ G_{31} & G_{33} & -H_{32} & -H_{34} \\ G_{41} & G_{43} & -H_{42} & -H_{44} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} \phi_{n,D}^1 \\ \phi_{n,D}^3 \\ \phi_N^2 \\ \phi_N^4 \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} H_{11} & H_{13} & -G_{12} & -G_{14} \\ H_{21} & H_{23} & -G_{22} & -G_{24} \\ H_{31} & H_{33} & -G_{32} & -G_{34} \\ H_{41} & H_{43} & -G_{42} & -G_{44} \end{bmatrix} \begin{bmatrix} \bar{\phi}_D^1 \\ \bar{\phi}_D^3 \\ \bar{\phi}_{n,N}^2 \\ \bar{\phi}_{n,N}^4 \end{bmatrix}}_{b} \tag{37}$$

More generally, it is possible to rewrite the $\boldsymbol{H\phi} = \boldsymbol{G\phi}$ matrix system as

$$\underbrace{\begin{bmatrix} \boldsymbol{G}_D & -\boldsymbol{H}_N \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{Bmatrix} \{\boldsymbol{u}\}_D \\ \{\boldsymbol{u}_n\}_N \end{Bmatrix}}_{\boldsymbol{x}} = \underbrace{\underbrace{\begin{bmatrix} \boldsymbol{H}_D & -\boldsymbol{G}_N \end{bmatrix}}_{\boldsymbol{B}} \underbrace{\begin{Bmatrix} \{\boldsymbol{u}\}_D \\ \{\boldsymbol{u}_n\}_N \end{Bmatrix}}_{\boldsymbol{c}}}_{\boldsymbol{b}} \tag{38}$$

where all the unknowns have been gathered in the left-hand side $\boldsymbol{x}$ vector which is premultiplied by the $\boldsymbol{A}$ matrix. The right-hand side of the equality can be computed to form the $\boldsymbol{b}$ vector. This forms a linear system that can be solved in order to know the full values of the potential and its directional derivative along the boundary. The last step is to associate each computed value to the right element so that each element has a proper potential and directional derivative value.

This kind of organization of the linear system is the reason why the vector of elements should be sorted.

2.3 Implementation of the representation formula inside the domain

2.3.1 Computation of the potential

Once the potential and its directional derivative is known for each element, it is possible to calculate the value of the potential at any point inside the domain starting from the representation formula given by Equation 26. This equation can be discretized in a similar way than the boundary representation formula. One obtains

$$\phi(\mathbf{P}) = -\sum_{j=1}^{N} G_{ij}\phi_n^j + \sum_{j=1}^{N} H_{ij}\phi^j \tag{39}$$

$$= -\sum_{j=1}^{N} \left(\int_{\Gamma_j} \varphi(\mathbf{P}, \mathbf{q})\, ds_{\mathbf{q}} \right) \phi_n^j + \sum_{j=1}^{N} \left(\int_{\Gamma_j} \frac{\partial \varphi(\mathbf{P}, \mathbf{q})}{\partial \mathbf{n}_{\mathbf{q}}}\, ds_{\mathbf{q}} \right) \phi^j. \tag{40}$$

By injecting φ from Equation 24 inside the above equation, one can write

$$\phi(\mathbf{P}) = -\sum_{j=1}^{N}\left(\int_{\Gamma_j}\frac{1}{2\pi}\ln(r)\,ds_{\mathbf{q}}\right)\phi_n^j + \sum_{j=1}^{N}\left(\int_{\Gamma_j}\frac{1}{2\pi}\frac{\partial\ln(r)}{\partial\mathbf{n_q}}\,ds_{\mathbf{q}}\right)\phi^j, \tag{41}$$

$$\approx -\frac{1}{4\pi}\sum_{j=1}^{N} l_j\phi_n^j\sum_{k=1}^{N^{GP}}\ln(r(\varepsilon_k))w_k + \frac{1}{2\pi}\sum_{j=1}^{N}(\alpha_{j+1}-\alpha_j)\phi^j, \tag{42}$$

$$= \sum_{j=1}^{N}\left(\frac{\alpha_{j+1}-\alpha_j}{2\pi}\phi^j - \frac{l_j}{4\pi}\phi_n^j\sum_{k=1}^{N^{GP}}\ln(r(\varepsilon_k))w_k\right) \tag{43}$$

Thus, by fully knowing the value of the potential and its directional derivative along the boundary of the surface, it is possible to determine its value inside the domain.

In practice, the domain will also be meshed where one needs to access the potential, so that it can be computed at the position of the nodes of the domain (or the barycenter of each element which is easier). For the nodes on the boundary, one technique is to associate to it the value of the potential of the element to which they belong. The problem is that this potential is discontinuous across 2 elements and so the value of the potential at a boundary node is ambiguous.

2.3.2 Computation of the electric field

From the equation for the electric potential expressed through the BEM within the domain, it is possible to derive an equation for the electric field. The potential equation for any point in the domain is given by:

$$\phi(\mathbf{P}) = -\sum_{j=1}^{N}\underbrace{\left(\int_{\Gamma_j}\varphi(\mathbf{P},\mathbf{q})\,ds_{\mathbf{q}}\right)}_{G_j}\phi_n^j + \sum_{j=1}^{N}\underbrace{\left(\int_{\Gamma_j}\frac{\partial\varphi(\mathbf{P},\mathbf{q})}{\partial\mathbf{n_q}}\,ds_{\mathbf{q}}\right)}_{H_j}\phi^j. \tag{44}$$

The electric field can be calculated as the opposite of the gradient of the potential:

$$\boldsymbol{E}(\mathbf{P}) = -\nabla\phi(\mathbf{P}). \tag{45}$$

Thus, taking the gradient from the Equation 44, one can write:

$$\nabla_x\phi(\mathbf{P}) = -\sum_{j=1}^{N}\underbrace{\left(\int_{\Gamma_j}\nabla_x\varphi(\mathbf{P},\mathbf{q})\,ds_{\mathbf{q}}\right)}_{G'_j}\phi_n^j + \sum_{j=1}^{N}\underbrace{\left(\int_{\Gamma_j}\nabla_x\left(\frac{\partial\varphi(\mathbf{P},\mathbf{q})}{\partial\mathbf{n_q}}\right)\,ds_{\mathbf{q}}\right)}_{H'_j}\phi^j. \tag{46}$$

To avoid confusion in the calculations, the convention is chosen such that any point inside the domain is denoted by $\boldsymbol{P} = (x, y)$ while points on the boundary are denoted by $\boldsymbol{q} = (\xi, \nu)$. As a reminder, the fundamental solution of the Laplace equation and the distance of any point in the domain from the boundary are written as:

$$\varphi(x,y) = \frac{1}{2\pi}\ln r \qquad\qquad r = \sqrt{(\xi-x)^2+(\eta-y)^2} \tag{47}$$

To facilitate the calculation of the derivatives of the H and G matrices, it is interesting to calculate the derivatives with respect to the spatial variables and to introduce the derivation formula with respect to the normal or tangential component:

$$\begin{cases} \dfrac{\partial r}{\partial x} = -\dfrac{\partial r}{\partial \xi} = -\dfrac{\xi - x}{r} = -\cos\alpha \\ \dfrac{\partial r}{\partial y} = -\dfrac{\partial r}{\partial \eta} = -\dfrac{\eta - y}{r} = -\sin\alpha \end{cases} \qquad \begin{cases} \dfrac{\partial r}{\partial \mathbf{t}} = \nabla r \cdot \mathbf{t} = -\dfrac{\partial r}{\partial \xi} n_y + \dfrac{\partial r}{\partial \eta} n_x \\ \dfrac{\partial r}{\partial \mathbf{n}} = \nabla r \cdot \mathbf{n} = \dfrac{\partial r}{\partial \xi} n_x + \dfrac{\partial r}{\partial \eta} n_y \end{cases} \tag{48}$$

It is thus possible to calculate the matrix G'_j, the integrand can be calculated as:

$$G'_j = \nabla_x \varphi(\mathbf{P}, \mathbf{q}) = -\nabla_\xi \varphi(\mathbf{P}, \mathbf{q}) = \begin{cases} \dfrac{\partial \varphi}{\partial x} = -\dfrac{\partial \varphi}{\partial \xi} = \dfrac{1}{2r\pi} \dfrac{\partial r}{\partial x} = \dfrac{x - \xi}{2\pi r^2} \\ \dfrac{\partial \varphi}{\partial y} = -\dfrac{\partial \varphi}{\partial \eta} = \dfrac{1}{2r\pi} \dfrac{\partial r}{\partial y} = \dfrac{y - \eta}{2\pi r^2} \end{cases} \tag{49}$$

The matrix H'_j requires more advanced calculations. Using the relations introduced earlier, it is possible to show that the integrant of H'_j can be written as:

$$H'_j = \nabla_x \left(\frac{\partial \varphi(\mathbf{P}, \mathbf{q})}{\partial \mathbf{n}_q} \right) = \nabla_x \left(\nabla_\xi \varphi(\mathbf{P}, \mathbf{q}) \cdot \mathbf{n_q} \right) = \begin{cases} \dfrac{\partial}{\partial x} \left(\dfrac{\xi - x}{2\pi r^2} \right) n_x + \dfrac{\partial}{\partial x} \left(\dfrac{\eta - y}{2\pi r^2} \right) n_y \\ \dfrac{\partial}{\partial y} \left(\dfrac{\xi - x}{2\pi r^2} \right) n_x + \dfrac{\partial}{\partial y} \left(\dfrac{\eta - y}{2\pi r^2} \right) n_y \end{cases} \tag{50}$$

$$= \begin{cases} \dfrac{(x - \xi)^2 - \frac{1}{2} r^2}{\pi r^4} n_x + \dfrac{(x - \xi)(y - \eta)}{\pi r^4} n_y \\ \dfrac{(x - \xi)(y - \eta)}{\pi r^4} n_x + \dfrac{(y - \eta)^2 - \frac{1}{2} r^2}{\pi r^4} n_y \end{cases} \tag{51}$$

In order to proceed to the integration of the integrants of the matrices H'_j and G'_j previously obtained, it is necessary to introduce a parametrisation of the element on the boundary.

$$\begin{cases} \xi(\varepsilon) & = \dfrac{x_{j+1} + x_j}{2} + \dfrac{x_{j+1} - x_j}{2} \varepsilon, \\ \eta(\varepsilon) & = \dfrac{y_{j+1} + y_j}{2} + \dfrac{y_{j+1} - y_j}{2} \varepsilon. \end{cases} \tag{52}$$

Finally, the gradient of the matrices H and G is obtained. It is possible to express the integral on the j element using the parametrisation developed previously. Then, the resolution of this

integral can be considered from a numerical point of view thanks to the Gauss quadrature.

$$\frac{\partial G_{ij}}{\partial x} = \int_{\Gamma_j} \frac{x_i - \xi}{2\pi r^2}\, ds_q = \int_{-1}^{1} \frac{x_i - \xi(\varepsilon)}{2\pi r^2(\varepsilon)}\, d\varepsilon \approx \sum_{k=1}^{N^{GP}} \frac{x_i - \xi(\varepsilon_k)}{2\pi r^2(\varepsilon_k)} w_k, \tag{53}$$

$$\frac{\partial G_{ij}}{\partial y} = \int_{\Gamma_j} \frac{y_i - \eta}{2\pi r^2}\, ds_q = \int_{-1}^{1} \frac{y_i - \eta(\varepsilon)}{2\pi r^2(\varepsilon)}\, d\varepsilon \approx \sum_{k=1}^{N^{GP}} \frac{y_i - \eta(\varepsilon_k)}{2\pi r^2(\varepsilon_k)} w_k, \tag{54}$$

$$\frac{\partial H_{ij}}{\partial x} = \int_{\Gamma_j} \frac{(x_i - \xi)^2 - \frac{1}{2}r^2}{\pi r^4} n_x^i + \frac{(x - \xi)(y_i - \eta)}{\pi r^4} n_y^i\, ds_q \tag{55}$$

$$= \int_{-1}^{1} \frac{(x_i - \xi(\varepsilon))^2 - \frac{1}{2}r(\varepsilon)^2}{\pi r(\varepsilon)^4} n_x^i + \frac{(x_i - \xi(\varepsilon))(y_i - \eta(\varepsilon))}{\pi r(\varepsilon)^4} n_y^i\, d\varepsilon, \tag{56}$$

$$\approx \sum_{k=1}^{N^{GP}} \left(\frac{(x_i - \xi(\varepsilon_k))^2 - \frac{1}{2}r(\varepsilon_k)^2}{\pi r(\varepsilon_k)^4} n_x^i + \frac{(x_i - \xi(\varepsilon_k))(y_i - \eta(\varepsilon_k))}{\pi r(\varepsilon_k)^4} n_y^i \right) w_k, \tag{57}$$

$$\frac{\partial H_{ij}}{\partial y} = \int_{\Gamma_j} \frac{(x_i - \xi)(y_i - \eta)}{\pi r^4} n_x^i + \frac{(y_i - \eta)^2 - \frac{1}{2}r^2}{\pi r^4} n_y^i\, ds_q, \tag{58}$$

$$= \int_{-1}^{1} \frac{(x_i - \xi(\varepsilon))(y_i - \eta(\varepsilon))}{\pi r^4(\varepsilon)} n_x^i + \frac{(y_i - \eta(\varepsilon))^2 - \frac{1}{2}r^2(\varepsilon)}{\pi r^4(\varepsilon)} n_y^i\, d\varepsilon, \tag{59}$$

$$\approx \sum_{k=1}^{N^{GP}} \left(\frac{(x_i - \xi(\varepsilon_k))(y_i - \eta(\varepsilon_k))}{\pi r^4(\varepsilon_k)} n_x^i + \frac{(y - \eta(\varepsilon_k))^2 - \frac{1}{2}r^2(\varepsilon_k)}{\pi r^4(\varepsilon_k)} n_y^i \right) w_k, \tag{60}$$

with i denoting the point in the domain at which the electric field has to be calculated.

The above computations show that a representation formula for the electric field can be derived and it can be integrated to the code by performing numerical integrations as for the G_{ij} components for the potential computation.

2.4 Numerical implementation of the visualisation

There are three ways for visualizing the data in the domain (through `GMSH`):

- The `ElementData` visualization type, which assigns a value to each element and thus displays a constant value over each element,
- the `ElementNodeData` visualization type, which assigns a value to each node of each element so that the field is interpolated inside each element, but can still be discontinuous across elements,
- and the `NodeData` visualization type, which assigns a value to each node and thus interpolates the value over all the domain.

Only the two first method have been implemented and used here. These implementations are described here below. In fact, the `NodeData` visualization type implies a unique value of the potential for each node which is wrong by construction on the boundary nodes. Also, the resulting displayed field would have been non-constant over the boundary elements (if the average value of the potential between two elements was chosen as the nodal value for example), which is truly in contraction with the BEM hypothesis.

2.4.1 Visualization by element

Once the solution of the potential has been calculated by the BEM along the boundary, it is possible to evaluate the potential for any point in the domain. Thus, even if the potential is discontinuous between two elements of the boundary, the smoothing effect of the Green's function allows to obtain a perfectly continuously differentiable potential within the domain. The `ElementData` visualization consists in assigning a constant value of the potential for each element of the domain. The point where the potential is evaluated is the barycenter of each element.

The advantage of this method is its ease of implementation. As the barycentre of an element lies strictly inside the BEM domain, the potential can be computed using Equation 43. As the computation of the potential at the barycentre of each element is independent from the other domain elements, it is easy to parallelize it (using `OpenmMP` in this case) to optimize the calculation time. Once the potential has been computed at the barycenter of each element, it is enough to associate the potential with the tag of the considered element and to provide the results to `GMSH` to visualize them.

However, this method leads to visualization discontinuities of the potential at the boundaries between the elements, whereas the potential should be smooth inside the domain.

The above method always refers to the computation of the potential, but this is also valid for visualizing the electric field, applying the exact same method as above.

2.4.2 Visualization by element nodes

In order to obtain a more realistic visualization and to observe the smooth and continuous behaviour of the potential inside the domain, the `ElementNodeData` visualization type has also been implemented.

In fact, since the potential has to be assigned to each node of each element, it will appear smooth inside the domain, but constant over each boundary element and thus discontinuous on the boundary. Since this is what represents the best the nature of the potential, this the preferred visualization method.

Although the association of the potential to each node of each element strictly included inside the domain is more difficult than for the `ElementData` visualization, this stays quite simple. In fact one simply needs to compute only once the potential at each node inside the domain and associates it with the node tag (under the form of a map in this case). Then, looping over the elements of the domain and their nodes, one can assign the proper potential to each element node.

What is more involved is the choice (or the computation) of the potential on the nodes of the boundary and mainly the association of the nodes to the proper elements.

The first part of the algorithm consists in retrieving the tag of all the elements having at least one node in common with the boundary. Normally, it would be necessary to loop over all the nodes of the elements of the domain and all the nodes of the boundary to find a correspondence. In order to optimize this step, the `getElementsByCoordinates` function of `GMSH` was used. Since this function returns the tags of the elements at a given point, one can only loop over the boundary nodes and retrieve the tag of the element in contact with the

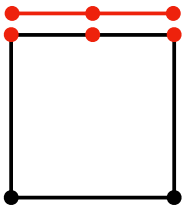
boundary. This truly improves the performance of the solver.

The second step is to loop over all these elements in contact with the boundary, then for each of their node, only focus on the nodes in contact with the boundary. One should then retrieve the tags of the elements in common with these nodes. In order to do that, before, one can construct a map assigning a vector of elements tags to each node of the boundary by looping over the boundary elements. By doing so, one can take advantage of the map data structure that is a lot more optimized than just looping over all the elements of the boundary every time. Several configurations can be observed:

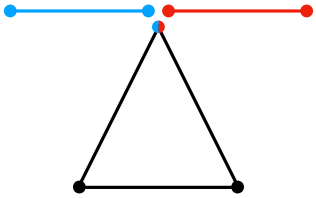
- the node is a mid element node and the value of the potential is simply the one of the unique "neighbour" element,
- the two neighbouring boundary element are not part of the domain element and the potential at that node is ambiguous. The mean of the potential of the two boundary element is thus assign to that node,
- only one of the two neighbouring boundary elements is part of the domain element and the potential at the node is simply the value of the potential of that element,
- the two neighbouring elements are part of the domain element and the potential at that point is again ambiguous. The potential is thus again computed as the average of the potential between the two boundary elements.

The result of this method is that the potential inside the domain appears smooth and the potential on the boundary appears discontinuous and constant over each element, which is more representative of the true nature of the potential. The only errors that it brings is on the boundary where the potential is ambiguous for certain nodes.

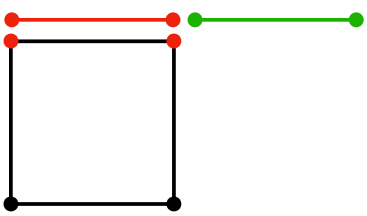
Contrarily to the `ElementData` visualization type, the `ElementNodeData` visualization type has not been implemented for the electric field.



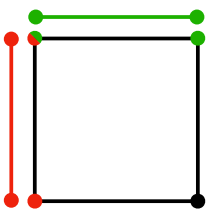
(a) Association of the potential for the midnode in the case of second-order elements.



(b) Association of the potential for a node at the discontinuity between two elements.



(c) Association of the potential for two nodes matching with a single element.



(d) Association of the potential for the three nodes matching two elements in a corner.

Figure 15: Different possible geometries for the encounter of a domain element with the boundary.

2.5 BEM singularity

One of the biggest weaknesses of the BEM is to be able to correctly evaluate the integrals when the boundary elements are close to each other. The purpose of this section is to explain the characteristic singularity of the BEM and how it has been handled in our implementation. The fundamental solution of the Laplace equation is given by Equation 24. The latter has a logarithmic singularity in zero. This is not a problem for the computation of the matrix elements H_{ii}, G_{ii} and H_{ij} since these are evaluated analytically. However, the calculation of G_{ij} is done numerically and therefore it is important to deal with the case of this singularity. A numerical integration uses Gauss points for evaluating the integral. When the integration element j is far from the singularity, the integration can be performed using a small number of Gauss points and still have a satisfactory approximation to the solution. Conversely, when the integration element j is close to the singularity, it undergoes large potential variations along its length. Gaussian integration will then require increasing the number of integration points in order to obtain a satisfactory result. This phenomenon can be seen in Figure 16. The singularity lies on the integration element i. The j_1 element is close to the singularity and will require a large number of Gauss points while the j_2 element is far away enough and will only need a small number of Gauss points for a good approximation of the solution. The problem associated with this singularity was all the more striking in the calculation of the electric field. The computation of the $\nabla\mathbf{H}$ and $\nabla\mathbf{G}$ matrices is performed through numerical integration and the electric field greatly diverges at points close to the boundary if the number of Gauss points is not sufficiently large. It should be noted that the number of integration points has a strong influence on the computation time and is is therefore important to minimize their number while ensuring a good approximation of the solution.

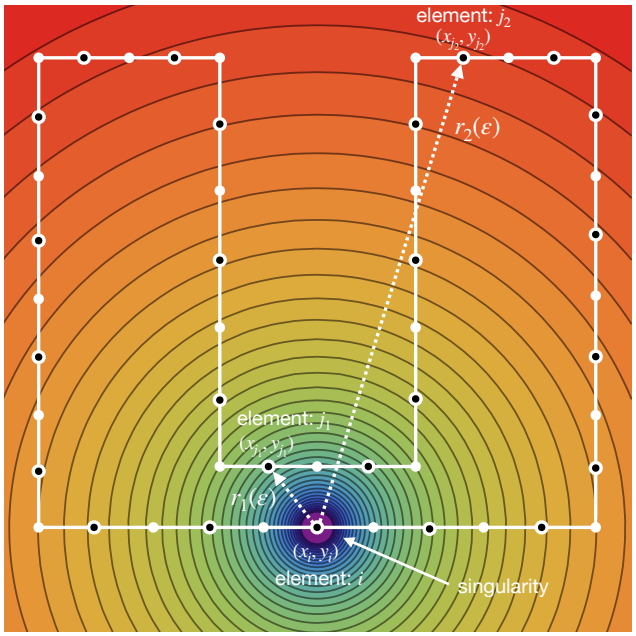


Figure 16: Figure of the fundamental solution of the potential and its logarithmic growth as well as the singularity at the midpoint of the integration element.

Different approaches have been considered to solve this problem of numerical integration and the treatment of these singularities. It was found out that the computation of the integrals of the BEM in a two-dimensional case could be computed analytically. However, it would have been very difficult to perform such integrations by hand. Consequently, the

software `Mathematica` was used in order to find the different analytical solutions. These could be computed both for the $\nabla\mathbf{G}$ and $\nabla\mathbf{H}$ matrices but also for the $\mathbf{G}$ matrix used in the calculation of the potential.

The main advantage of the analytical solutions was the decrease in the computation time of the code while providing much more accurate results. In fact, numerical integration near a singularity requires a lot of Gauss points, resources and computation time, especially since the Gaussian integration is not optimal for dealing with that kind of problem. Moreover, since the analytical solutions avoid dealing with singularities of integrands, this is also more robust to different geometries and meshes (even when the element are relatively close to each other).

A convergence study of the numerical solution (towards the analytical solution) as a function of the number of Gauss points, will be carried out in the rest of this report.

The differences in computation time between the two method is introduced below.

The `rectangle.geo` file where the length and width were fixed to the unit and subdivided into 100 elements. The calculation of the matrix $\mathbf{G}$ appears in the resolution of the potential on the boundary elements and in the calculation of the potential inside the domain. The calculation of $\nabla\mathbf{G}$ and $\nabla\mathbf{H}$ only appears in the evaluation of the electric field.

CPU time for the numerical solution			
Number of Gauss points	computeInternalPhi [ms]	computeElementData [ms]	total time of the code BEM [ms]
2	1231.62	9557.42	11348.4
4	1534.72	11945.1	14049.1
6	1683.04	13289.4	15557.3
8	1866.22	14611.2	17083.8
10	1933.07	15573.9	18100.6
12	2084.77	17226	19928.3
14	2140.04	17857	20601.8
16	2296.15	19475.7	22391.6
18	2846.67	21959.6	25441.6
20	2881.35	24539.3	28365.4
22	2725.68	23422.5	26780.8
24	2815.54	24669	28131.5
26	2813.49	27785.7	31213.9
28	3183.78	27620.6	31450.5
30	3033.35	30274	33944.1
analytical	448.804	1615.84	2618.44|

Table 1: Analysis of the CPU time [s] as a function of the number of Gauss points used for integration and comparison with the CPU time [s] of the analytical solution.

The comparison of the performance between the analytical solution and the numerical solution is immediate and is displayed in Tab. 1. The analytical solution is much faster than the numerical one, while being more accurate. It should be stressed that it is necessary to use at least a dozen Gauss points to ensure a good convergence of the electric field. The difference in performance then exceeds a factor of ten in this configuration. If the integration is pushed to the 30 Gauss points that correspond to the maximum allowed by `GMSH`, then the difference

with the analytical solution reaches almost a factor 20.

Based on this table, a graph of the total BEM solver computation time versus the number of Gauss points used can be seen in Figure 17.

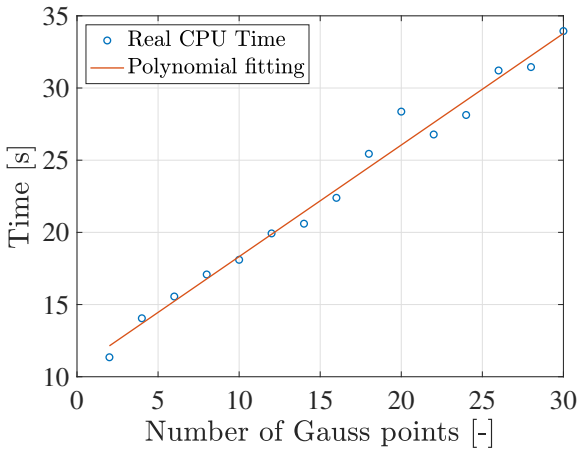
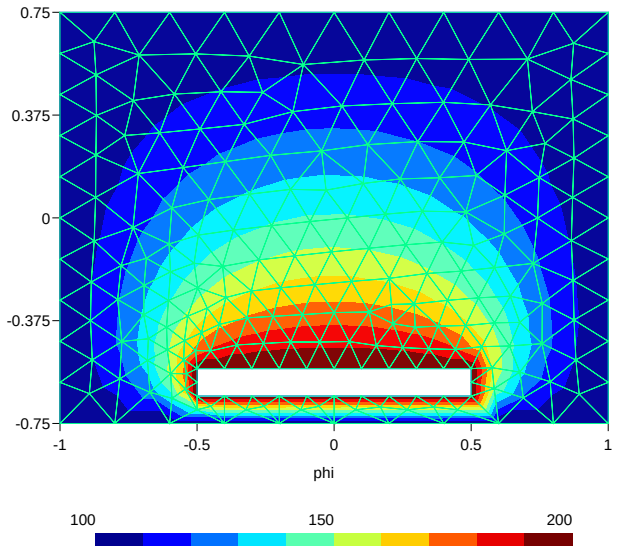


Figure 17: CPU time as a function of the number of Gauss points used in the integration considering the total time of the BEM code.

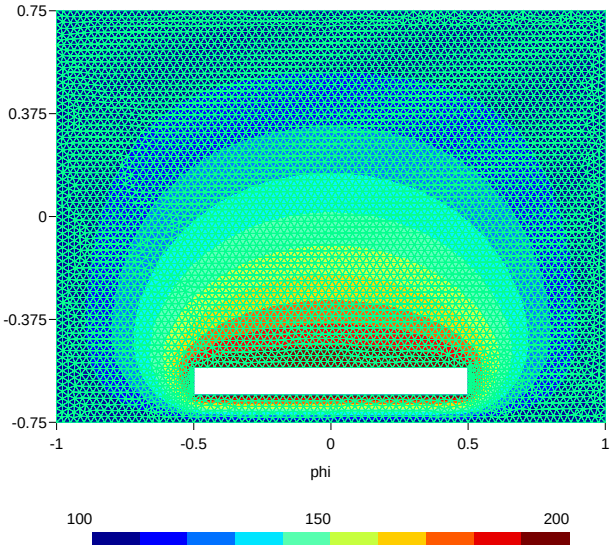
A linear regression model could be fitted to the calculation time. As the number of Gauss points is used for the integration of the boundary elements, it is normal to observe a linear complexity of the integration time.

2.6 First results

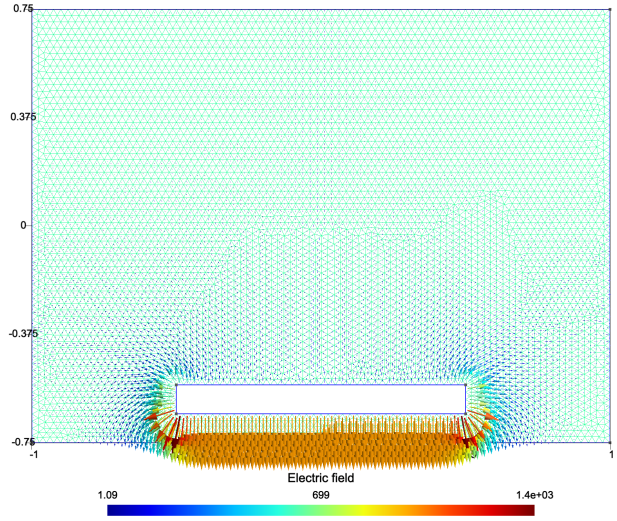
A Figure illustrating the results obtained with the above method is shown on Figure 18. In order to see the influence of the mesh, one opted for 10 elements on the edge in the first case and 80 elements along x and 60 along y in the second.



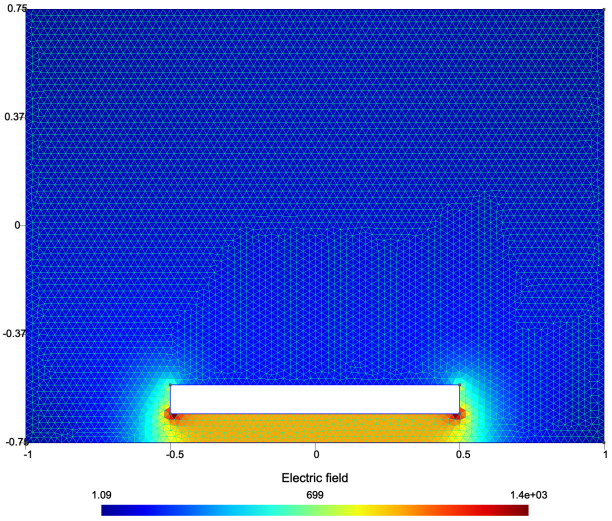
(a) Potential with 10 elements by edge.



(b) Potential with 80 and 60 elements by edge.



(c) Electric field in vector form with 80 and 60 elements on the edges.



(d) Norm of the electric field with 80 and 60 elements on the edges.

Figure 18: Electric potential and electric field in a domain with a hole described in the `hole.geo` file. The outer boundary of the domain was set at 100 V while the boundary of the hole inside the domain was set at 200 V

The result seems to be consistent with what is expected in such a situation. The potential seems to evolve from the value set at the outer boundary to the value at the hole in a smooth, continuous and homogeneous way. Also the electric field seems to be constant below the hole, which is in accordance with the parallel plates capacitor.

However, the colour scale proposed for the visualisation in `GMSH` does not always allow to understand the behaviour of the potential. Figure 19 shows the electric potential on a cut in the domain. The purpose of this slice is to display the evolution of the potential in a more quantitative manner. This graph was made using the `GMSH` plugin `CutPlane`. A simple linear interpolation was performed between the different points to obtain the final result.

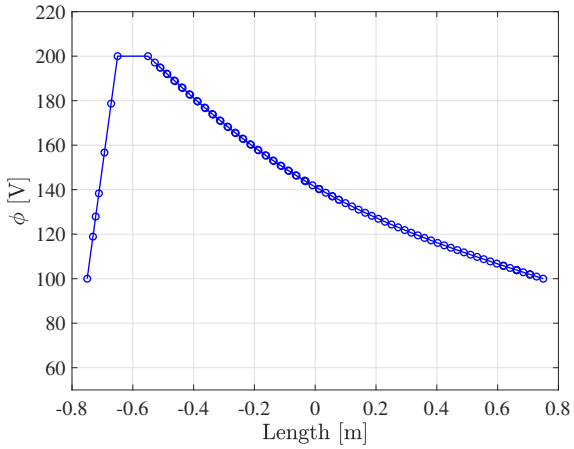
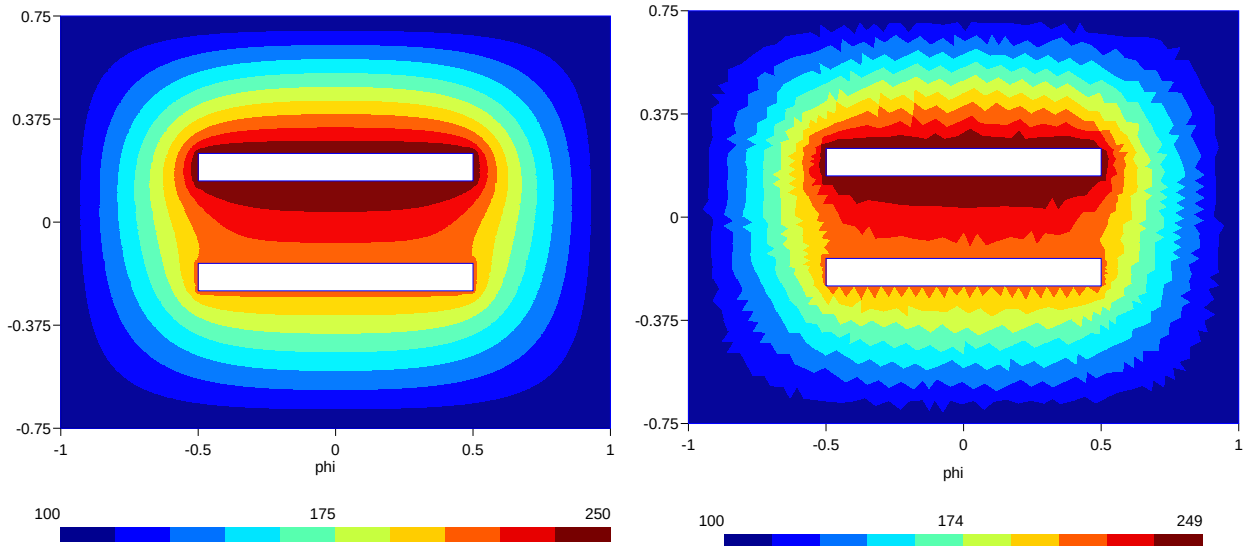


Figure 19: Graph of the potential for a cut located in the middle of the file `Hole.geo` with 200 [V] imposed on the hole and 100 [V] imposed on the boundary.

Finally, a comparison of the two visualization methods implemented is shown in Figure 20.

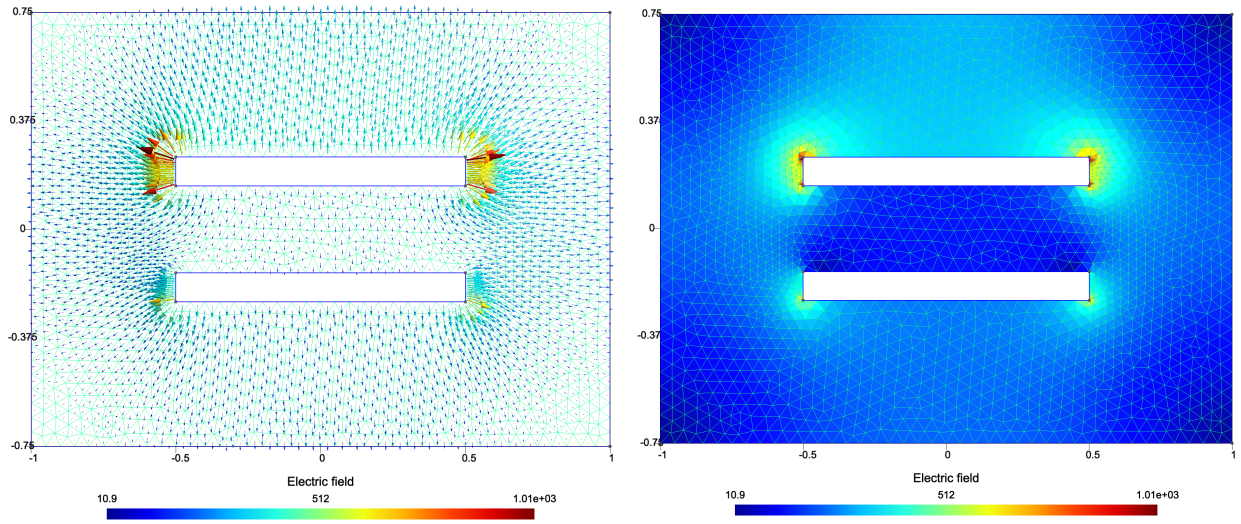


(a) Visualisation using the `ElementNodeData` visualization type of `GMSH`.

(b) Visualisation using the `ElementData` visualization type of `GMSH`.

Figure 20: Comparison of the two visualizations implemented for the BEM solver on the `doubleHole.geo` configuration with the potential fixed to 250 [V], 210 [V] and 100 [V] on the upper hole, bottom hole and boundary respectively.

The electric field of the same configuration can be seen in Figure 21.



(a) Electric field in vector form.

(b) Norm of the electric field vector.

Figure 21: Electric field for the geometry of the `doubleHole.geo` file with the same Dirichlet boundary conditions as in Figure 20.

As in Figure 19, Figure 22 shows the potential on a slice of the `doubleHole.geo` file with the same Dirichlet boundary conditions as in Figure 20.

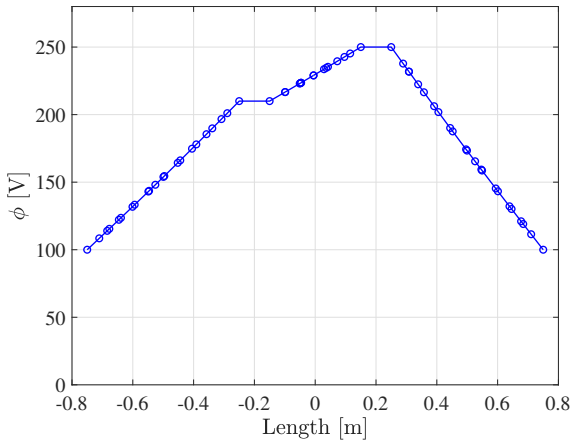
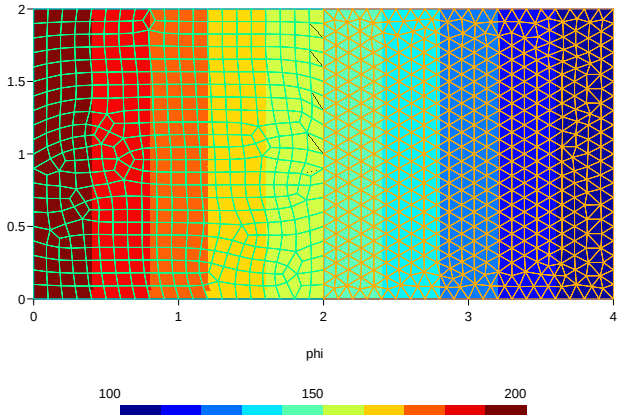
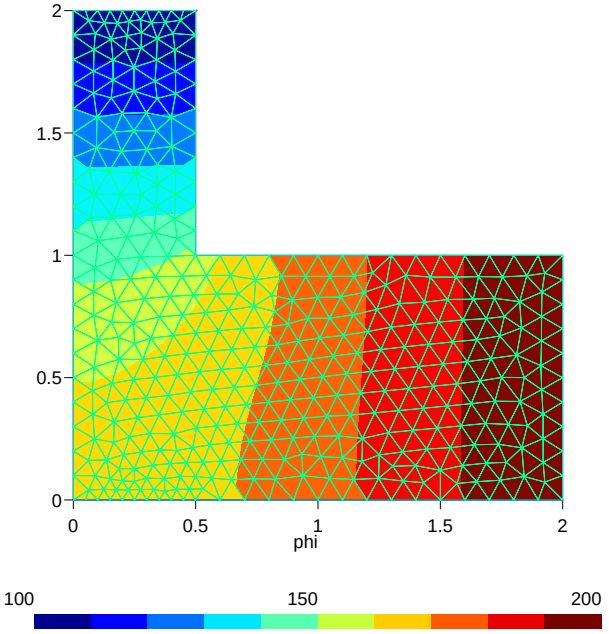


Figure 22: Graph of the potential for a cut located in the middle of the geometry described by the file `DoubleHole.geo`.

Different geometries have also been tested. A study of concave shapes is displayed in Figure 23b. In addition, a geometry consisting of two distinct meshes with second-order elements was considered in order to check the robustness of the code. This configuration can be seen in Figure 23a.



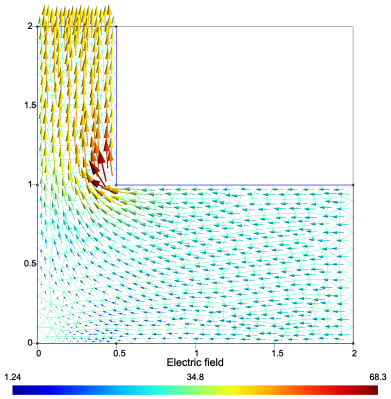
(a) Geometry consisting of two types of meshes. The first type of elements are squares while the second type of elements are triangles, both of order two. This is described by the `complex_validation.geo` file.



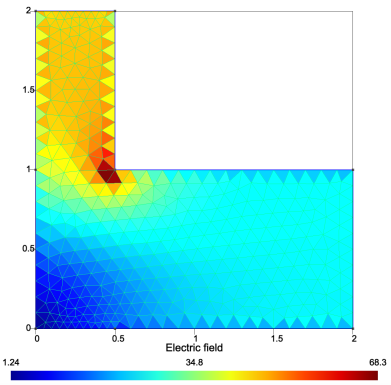
(b) Geometry with a concave shape described by the `l_shape.geo` file.

Figure 23: Application of the boundary element method to more complex meshes.

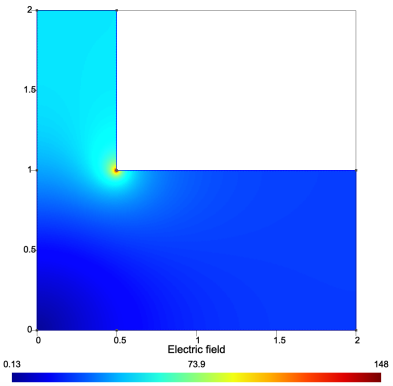
The electric field associated to the `l_shape.geo` file was also tested. The fact that it is a concave geometry with a right angle generates a singularity for the electric field. This singularity becomes more visible as the mesh is refined, as can be seen in Figure 24. The mesh was removed in this image because the potential was barely visible.



(a) Displays the electric field in vector form.



(b) Display of the electric field norm.



(c) Display of the electric field norm with a refined mesh.

Figure 24: Electric field diagram for the geometry described by the `l_shape.geo` geometry file.

2.7 Multiple BEM domain

The ultimate test of the BEM code was to be able to manage a separate domain in several parts. As the problem is based on the application of an electrostatic potential, it seemed appropriate to split the domain to simulate the impact of two electrodes for example. It

should be noted that this difficulty is mainly due to the two-dimensional nature of the problem. If the problem had been three-dimensional, the different 2D domains could certainly have been connected by a common volume. Considering the construction of `GMSH` a first idea could be to simply divide the domain by gathering them within various `PhysicalGroups`. However, such an option is in reality difficult to implement due to the fact that the different BEM domains must be independent. This is particularly problematic when applying boundary conditions. Several tests have been performed. It is not easy to show graphically the problem with this method but errors were found in the results. This was particularly true when the different BEM domains were close to each other. Irregularities in the values of the boundary potential were observed. It is possible that a detail was missed in the use of this `GMSH` tool. To avoid any problems, the decision was made to consider each BEM domain independently. Thus, for each BEM domain appearing in a geometry file, the BEM code will be called iteratively in order to solve them one after the other. It is then possible to group the different potentials and electric fields in a single view for display. This method is not limited to two BEM domains and can therefore be used for any number of these domains. The implementation of this functionality will be heavily used for the applications that will be presented in the following. A simple example of the division of BEM domains is shown in Fig. 25. This consists of two BEM domains separated by a beam embedded on either side. On each of the two domains, Dirichlet and Neumann conditions are defined on opposite sides. The first domain has Dirichlet conditions on the top and bottom with Neumann conditions on the right and left edges, while the second domain has Neumann conditions on the top and bottom with Dirichlet conditions on the right and left. This geometry is taken from the file `two_BEM_test.geo`.

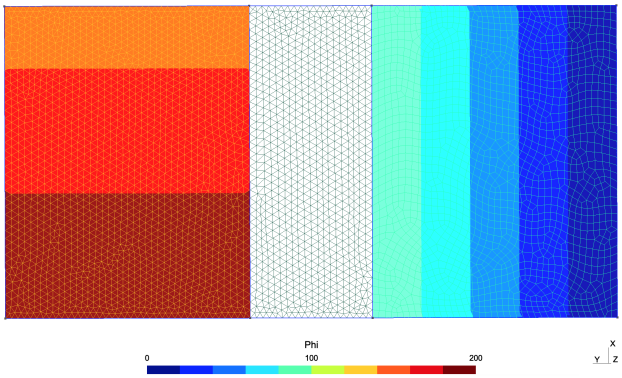


Figure 25: Simple geometry from the file `two_BEM_test.geo` illustrating the use of two BEM domains.

2.8 Performance and parallelisation

With the aim of improving computational performance, the most time-consuming routines and functions in the code were analysed.

The first most time consuming part is solving the linear equation system. Tough, it is very difficult to do anything about it as it is solved using the solver in the `Eigen` library of `C++`. Several `Eigen` solvers were tested in order to increase the resolution speed and performance of the code. It turned out that the `partialPivLu` solver was the most suitable for the BEM code. A more detailed study of the performance of the solvers is described in the next subsection. This will allow to justify this choice in a more rigorous way.

The second and most resource-intensive part is the calculation of the potential inside the domain. Fortunately, the BEM code has the advantage of being easily parallelizable. The calculation of the potential at the barycentres or nodes can be done independently and the problem is embarrassingly parallelizable. Note however that this is less trivial for the `ElementNodeData` visualization type.

OpenMP was also applied to the calculation of the elements of the matrices $\boldsymbol{G}$ and $\boldsymbol{H}$. This part of the code is not the most time-consuming compared to the previous ones. It is therefore difficult to see a significant improvement in the total code time.

For the sake of exercise rather than real performance improvements, OpenMP has also been implemented on other smaller code functions. As these functions participate in an insignificant percentage of the BEM code compared to those described above, no performance analysis was performed. It is quite possible that the use of OpenMP in such situations is counterproductive, but again this is negligible compared to the computation time of the main routines.

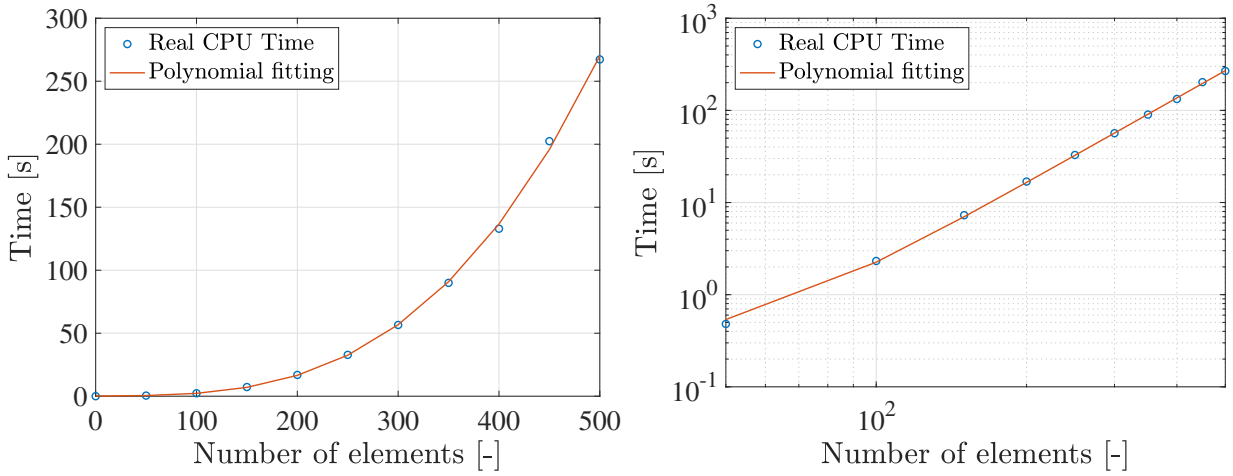
2.9 Complexity of algorithms

In this section, a study of CPU time will be carried out. It is important to specify that the execution time of the program is obviously dependent on the components of the computer on which it is launched. The aim is above all to present average calculation time results for the execution of the BEM solver on a relatively recent machine. To do this, a geometry was set for performance testing. It was possible to play around with a large number of parameters. The tests were carried out so that the BEM solver was at the top of the capabilities it is likely to provide. The geometry used was the `rectangle.geo` where the length and width was set to unity. The number of elements was set to 100 in these two dimensions. To solve the linear equation system, the `partialPivLu` solver from the `Eigen` library was used. This one has the advantage of being compatible with OpenMp. The test was carried out using first-order triangles. The various calculations are performed using the analytical solution of the integrals, both for the potential and for the electric field. Parallelization using OpenMp was performed with a number of threads equal to 12. For this experiment, an idea came out to check the time of each main algorithms that compose the BEM. It was thus possible to observe their complexity as well as the final complexity of our solver. The results are displayed on the Tab. 2

CPU time for a square geometry											
Dimensions	50	100	150	200	250	300	350	400	450	500	complexity
fillElementVector [ms]	4.673	3.48806	4.55809	5.35798	5.81503	6.9139	8.40902	8.68392	10.7739	11.0459	$\mathcal{O}(n)$
fillSystem [ms]	7.3421	24.72	54.6901	111.632	171.674	231.721	318.085	387.932	496.784	585.168	$\mathcal{O}(n^2)$
Solve linear system [ms]	102.833	284.15	568.803	1197	2184.59	3480.08	5426.99	7643.24	10420.5	13547.8	$\mathcal{O}(n^{5/2})$
computeElectrostaticPressure [ms]	6.19292	4.28104	5.31411	6.51288	8.26597	9.5849	11.2312	12.7861	15.7599	17.725	$\mathcal{O}(n)$
getModel [ms]	16.5451	70.308	165.576	259.418	383.515	521.679	691.3	900.911	1128.18	1414.31	$\mathcal{O}(n^2)$
computeInternalPhi [ms]	81.6369	396.472	1267.63	3017.69	5992.78	10310.3	17109.4	25599.6	38218.8	54369.4	$\mathcal{O}(n^3)$
elementNodeData [ms]	41.5909	118.19	260.051	420.37	676.407	953.588	1176.22	1678	1967.36	2356.76	$\mathcal{O}(n^{1.8})$
computeElementData [ms]	201.94	1361.19	4823.74	11666.7	23107	40638.8	64726.6	96078.8	149318	193999	$\mathcal{O}(n^3)$
total time of the code BEM [ms]	481.054	2321.29	7280.73	16884.1	32797.8	56554.7	89968.3	132944	202419	267298	$\mathcal{O}(n^3)$

Table 2: CPU time of the different main BEM routines and highlighting the algorithms complexity.

The major result of this complexity study was to learn that the BEM solver has a total cubic complexity. This performance can be considered quite poor. However, it is possible to put this result into perspective. Indeed, most of the algorithms have a quadratic complexity. Only the calculation of the potential and the electric field inside the domain has a cubic complexity. These belong to the post-processing category of the solver. This detail is important for the following. In the non-linear FEM-BEM coupling, the BEM solver is called iteratively but only the solution on the boundary at the edge of the domain is required. The calculation of the potential and the electric field inside the domain is performed only once at the end of the iterations when the algorithm has converged. It is also interesting to ask why the computation of the potential and the electric field in the domain have a cubic complexity. If the length scale is doubled then the number of nodes and elements is quadrupled since a two-dimensional geometry is considered. However, for each node or barycentre of the domain where the calculation of the potential and the electric field is performed, the whole set of elements of the boundary must be traversed. This multiplies this quadratic complexity by one more dimension and is the reason for this cubic result which was found empirically. There are two particular complexities. The first concerns the `partialPivLu` of `Eigen` which has an exponent close to 2.5. This has been evaluated empirically. The same applies to the second complexity which concerns `elementNodeData` which has a complexity with an exponent close to 1.8. These results were obtained by minimizing the polynomial in the least squares sense in order to obtain the curve that best fit the results. Considering that the complexity of functions is polynomial is an assumption that may not always be verified. The case of more exotic complexities such as logarithmic [2] or more particular laws have not been considered. Fig 26 shows the total time of the BEM algorithm when refining the mesh. The data is superimposed on a cubic interpolation thus empirically highlighting the complexity of the algorithm.



(a) Linear scale.

(b) Logarithmic scale.

Figure 26: Graph of the calculation time of the BEM solver as a function of the number of elements used over the length and width of the `rectangle.geo` file. Interpolations of the empirical data by a polynomial of degree three.

To conclude this section on CPU time, a study on the solution time of the linear system $Ax = b$ has been performed. Several solution systems were available in the `Eigen` library.

[2] In reality, this domain has been explored but no relation to a logarithm could be established

The computation time was performed on the same geometry. As already presented previously, the file `rectangle.geo` was used. The number of elements was set to 100 for the length and width. These dimensions were both set to a unit size.

CPU time for different kind of solvers						
	partialPivLu	fullPivLu	householderQr	colPivHouseholderQr	fullPivHouseholderQr	completeOrthogonalDecomposition
time [ms]	248.74	1180.36	714.478	858.496	1408.56	831.698

Table 3: CPU time [s] of the various systems of linear equations solvers proposed by the `Eigen` library.

It can be seen on Tab. 3 that the `partialPivLu` solver is the fastest of all. This result can be explained by the fact that it is the only solver tested compatible with OpenMp. It is enough to specify to `Eigen` the presence of multi-threading during the execution of the code so that this one uses it among the algorithms which allow it. However, it should be noted that speed is not the only criterion for these solvers. Some of these solvers provide a more accurate solution than others. Precision is generally antagonistic to speed, which may explain the slowness of some algorithms. The use of `partialPivLu` provides a sufficiently accurate result for the problem. However, there is a condition for its use. The matrix $\mathbf{A}$ must be inversible. This condition has never been violated on all the tested geometries. Nevertheless, it is possible that a particular case was missed. This one could have put this hypothesis in default. If this situation were to occur, it should be advisable to switch to another solver without any precondition on the coefficient matrix.

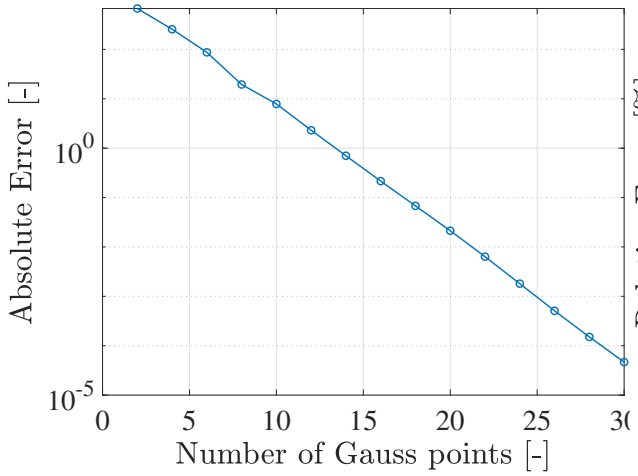
2.10 Convergence of the numerical solution

After having determined a numerical solution and an analytical solution for the calculation of the matrices $\nabla\mathbf{H}$, $\nabla\mathbf{G}$ and $\mathbf{G}$, it may be interesting to compare these two results. The idea is to study the convergence of the numerical solution with respect to the analytical one as a function of the number of Gauss points used for the integration. Moreover, only the calculation of the electric field will be analyzed. The aim is to show the order of the error and its convergence in order to extrapolate the result to a three-dimensional geometry where an analytical solution is no longer possible. The study will focus on the global error. This may vary according to the complexity of the geometry. In particular, integration on elements close to the singularity of the potential will provide a larger error than elements far from this singularity for the same number of Gauss points. The Figure 27a and Figure 27b show the relative and absolute error found via this analysis. These errors are calculated using the two norm of vector spaces. The absolute and relative errors are respectively given by:

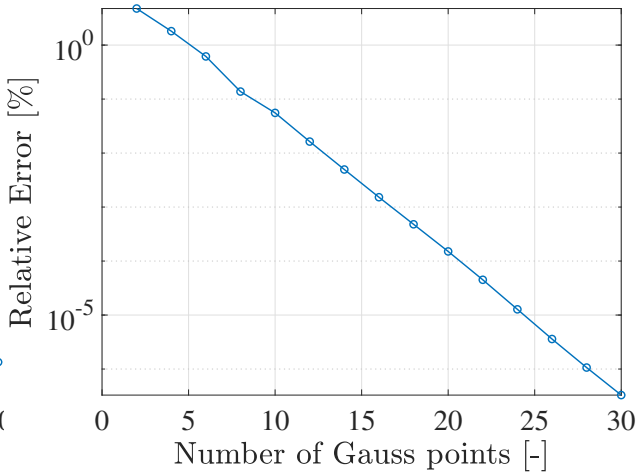
$$\varepsilon_{abs} = \|\hat{\mathbf{x}} - \mathbf{x}\| = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - x_i)^2}, \qquad \varepsilon_{rel} = \frac{\|\hat{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\hat{x}_i - x_i}{x_i}\right)^2}. \tag{61}$$

where in this case N is the total number of elements since the electric field is evaluated at the barycentre of each element. The vector $\hat{\mathbf{x}}$ is the set of electric field values for the numerical solution. While $\mathbf{x}$ is the vector of the values of the analytical solution of the electric field for all the elements.

For this first test, the geometry used is that of `rectangle.geo`. The width and length were set to the value of 1 and each of these dimensions was subdivided by 100 to create the mesh. This is composed of triangular elements of order 1. It was important to start with a simple geometry in order to validate the results. The maximum number of possible Gauss points with `GMSH` is 30. Thus, it was not possible to study the convergence beyond this limit. It should be noted that the use of specific functions such as the `ln` and the `atan` in the calculation of the analytical solution could eventually lead to rounding errors. One can observe the convergence of the numerical solution towards the analytical solution as a function of the number of Gauss points used. This confirms that the analytical solution found for this two-dimensional case is relevant. It is both more accurate and faster for this particular type of problem.

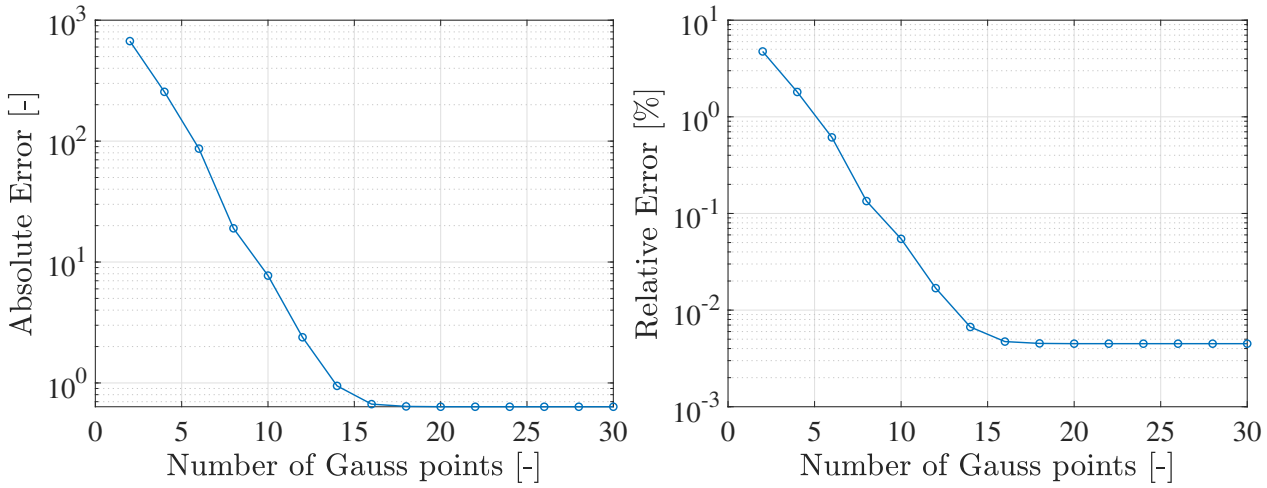


(a) Absolute error as a function of the number of Gaussian points for the file `rectangle.geo`.



(b) Relative error as a function of the number of Gaussian points for the file `rectangle.geo`.

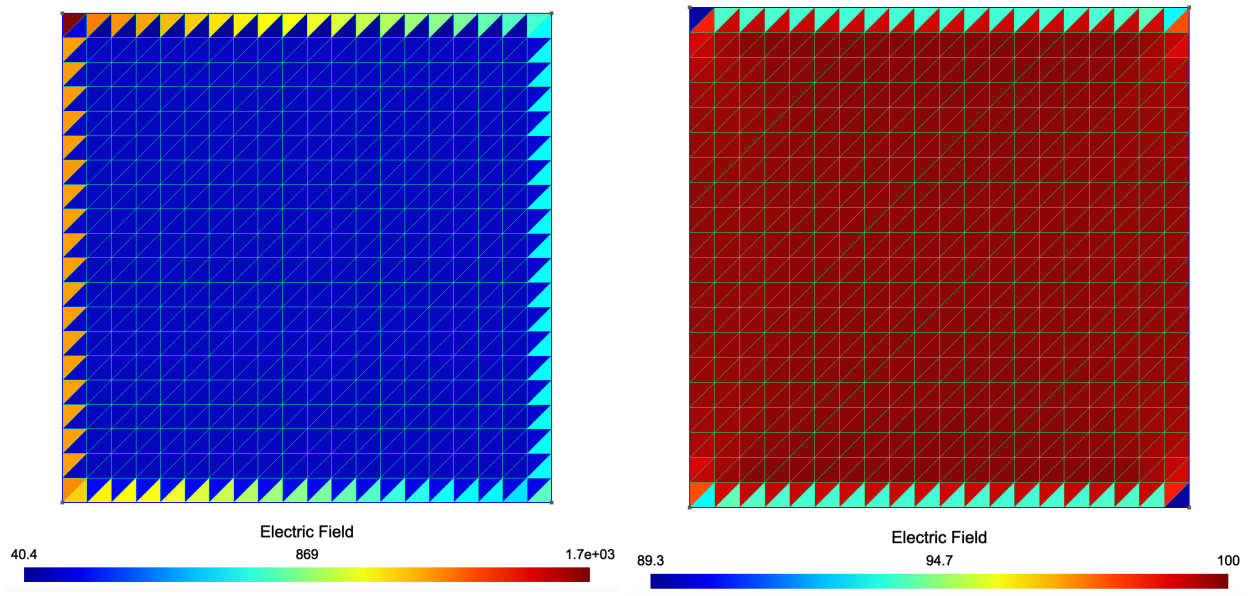
This study shows that the numerical solution is able to approach this analytical solution by increasing the number of Gauss points for the integration. However, it is important to highlight another problem. The analytical solution obtained is also an approximation. It comes from the evaluation of the integrals used for the calculation of the matrices $\nabla\mathbf{H}$, $\nabla\mathbf{G}$ and $\mathbf{G}$, but these are already derived from the discretisation of the problem by the BEM. The approximation of the potential by a constant value along the boundary implies an error with respect to the true analytical solution of the problem. In order to reduce this error, it is necessary to increase the number of discrete elements used. The effects of this error on the calculated electric field near the boundary have been observed in the case of the `retangle.geo` file, as there is a real analytical solution to the corresponding problem. This can be reduced to the study of a capacitor. Considering the application of a potential of 100 V on one side and 200 V on the opposite side, the electric field should reduce to a constant value of 100 [V.m] for all the elements. In the BEM approximation, this is not observed along the edges where an offset occurs. Thus, studying this time the convergence of the numerical solution as a function of the number of Gauss points with respect to this true analytical solution, it was found that the convergence stagnates after a certain number of Gauss points and that the error is consequently constant as depicted in Figure 28a and 28b.



(a) Absolute error as a function of the number of Gaussian points compared to the true analytical solution for the file `rectangle.geo`.

(b) Relative error as a function of the number of Gaussian points compared to the true analytical solution for the file `rectangle.geo`.

Figure 29a and 29b show the norm of the electric field for 2 and 30 Gauss points respectively. The 30-points solution is very close to the analytical solution. For better visualisation the number of elements on each side has been set to twenty. For an integration using two Gauss points, it can be directly observed that the electric field solution has a huge error. The solution with thirty Gauss points approximates the electric field very well inside the domain but suffers from an error along the edges. This is a direct result of the BEM discretisation and can be reduced by refining the mesh.

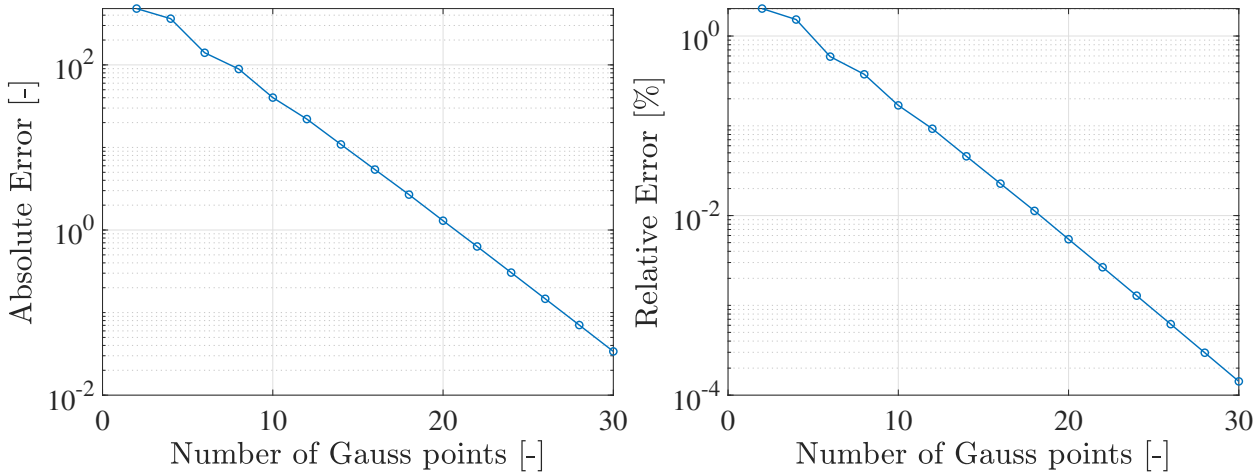


(a) (b)

Figure 29: Norm of the electric field for the numerical solution with an integration performed using respectively 2 (a) and 30 (b) Gauss points.

Finally, the convergence was tested on the geometry of `hole.geo` to show its validity against more complex models. The absolute and relative convergence for this example can be seen

in Figures 30a and 30b. The conclusions are identical to those drawn previously. For this case, it is not possible to study the offset with respect to the true analytical solution. Its existence can nevertheless be highlighted by studying the convergence of the error when the mesh is refined.



(a) Absolute error as a function of the number of Gaussian points for the file `hole.geo`.

(b) Relative error as a function of the number of Gaussian points for the file `hole.geo`.

2.11 Order of convergence

The convergence of the method can be visualized by comparing a numerical and an exact solution and by refining progressively the mesh.

More precisely, in each point at which the numerical solution was computed, it is possible to compare it with the exact solution and to build a local relative error. Then one can average this local error to compute a global error. The way the average is computed is

$$|\boldsymbol{x}| = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}, \tag{62}$$

where $|.|$ represents the average and $\boldsymbol{x}$ is a vector of size N.

Here the location of the points where the comparison between numerical and exact solution was made is the barycenter of each element.

The convergence of the method has been studied on the particular case of a square of unitary side length, with the potential being 100 [V] and 200 [V] on the left and right sides (respectively). Zero flux were applied on the top and bottom sides of the square. If the center of the coordinates system is located at the bottom-left corner of the square and if the axis are parallel to the sides of the square, the exact solution writes

$$\phi_{\text{exact}}(x, y) = 100 + (200 - 100)x = 100 + 100x = 100(1 + x) \text{ [V]}, \tag{63}$$

$$E_{\text{exact,x}}(x, y) = 100 \text{ [V/m]} \tag{64}$$

The results are presented in Figure 31. It can be seen that the electric field barely converges towards the analytical solution. This can be explained by the fact that the values of the

electric field at the boundaries is really bad, which implies a great error. In order not to take into account that boundary error, the same exact study has been performed for the electric field, but for a window inside the domain. This error is referred as the "Electric Field without edges" caption in the legend of Figure 31. This shows that apart from that side effect, the electric field well converges towards the analytical value.

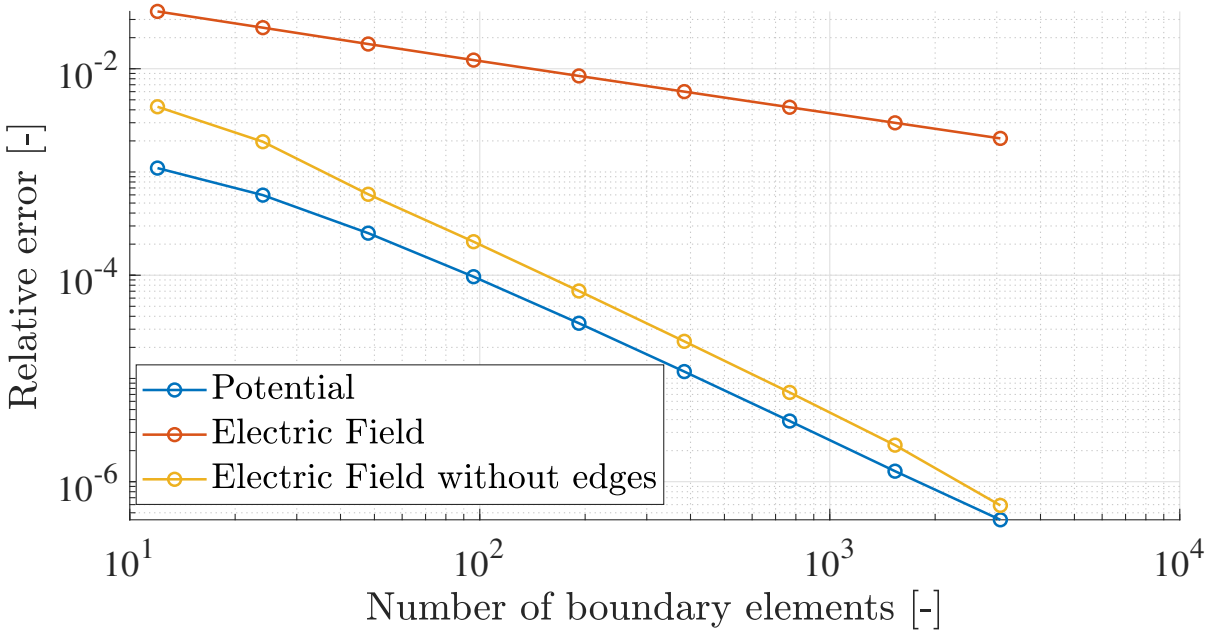


Figure 31: Study of the convergence of the BEM solver for a problem described by the `BEM_convergence.geo` file.

3 Linear FEM-BEM coupling

The coupling between the electrostatics and the kinematics of a structure is due to the electrostatic pressure resulting from the presence of an electric field on the boundary of a conducting surface. In this section, the one-way coupling between the BEM solver and the FEM solver is introduced. In Section 5, a more sophisticated two-way coupled solver is described.

3.1 Electrostatic pressure derivation

The first objective of the coupling is to compute analytically the electrostatic pressure applied on a charged electrode due to a potential difference between a mass electrode and the charged one.
The physical principle is the following one: the charged electrode carries a surface charge distribution so that the electric field due to the potential difference between two electrodes exertq a force on the charged surface. Since the electrode is considered as an ideal conductor, the charges are constrained to lie on its surface. Then, the electric field creates a pressure by its action on the surface charge, which eventually induces a displacement of the geometry.

To obtain the expression of the electrostatic pressure, it must be recalled that the normal component of the electric field E_n on both sides of an infinite charged plane is given by:

$$E_n = \frac{\sigma}{2\varepsilon}, \tag{65}$$

in which σ [C/m^2] denotes the charge per unit surface on the electrode and ε [F/m] the dielectric permittivity of the medium outside of the conductor.
However, the electrode is not an infinite surface, but if the electric field just above or just below an infinitesimal square of the electrodes surface is considered, one can assume that the electric field is evaluated at a very small distance from the square in comparison to the its dimensions. In that case, the electric field can be considered close to that of an infinite charged plane.

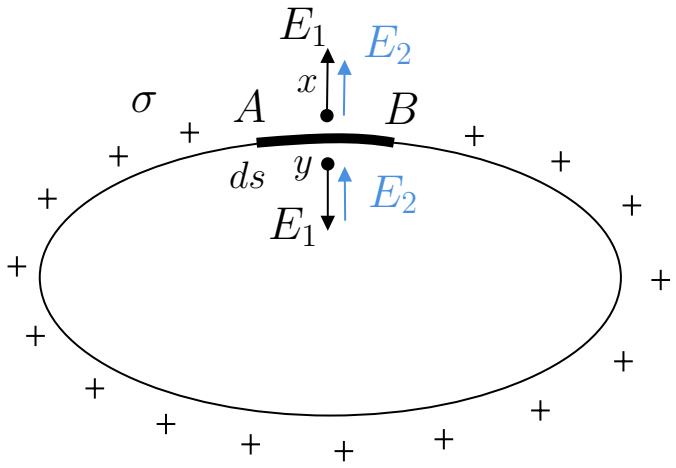


Figure 32: Electric fields experienced by a point P_x outside a conductor and a point P_y inside a conductor.

The configuration represented in Figure 32 is considered. In particular, the focus is set on an infinitesimal surface **AB** of the electrode, with its surface denoted as ds. Since the

charge density on the electrode is equal to σ, the total charge on the section **AB** is given by $dq = \sigma ds$. Two points are then considered: one point P_x outside the surface, but close to the section **AB** and one point P_y inside the surface and also close to the same section. Each point undergoes an electric field $\mathbf{E_1}$ due to the charge density of the section **AB** in the direction pointing out of the surface. Moreover, each point also undergoes an electric field $\mathbf{E_2}$ due to the charged density of the rest of the conductor. Since the two points P_x and P_y are incredibly close to the section **AB**, the field $\mathbf{E_2}$ is in the direction of the outside normal for both points, as shown in Figure 32.

Since the electric field outside a conductor (the charged electrode, in the present context) is given by $\sigma/2\varepsilon$, the net electric field at P_x is

$$E_x = E_1 + E_2 = \frac{\sigma}{\varepsilon}. \tag{66}$$

Furthermore, the electric field inside a conductor being equal to zero, the net electric field at P_y is

$$E_y = E_1 - E_2 = 0, \tag{67}$$

so that $E_1 = E_2$. Hence:

$$E_1 = E_2 = \frac{\sigma}{2\varepsilon}. \tag{68}$$

Knowing the electric field outside the charged electrode, the force experienced by the section **AB** is

$$F = dqE_2 = dq\frac{\sigma}{2\varepsilon} = \frac{\sigma^2}{2\varepsilon}ds. \tag{69}$$

Since the electric field outside the surface is given by $E = \sigma/\varepsilon$ the corresponding surface charge is $\sigma = E\varepsilon$.

Finally, the electrostatic pressure p_e [Pa] exerted on a charged surface with a normal component of the electric field E, is given by:

$$\boxed{p_e = \frac{F}{ds} = \frac{1}{2}\varepsilon E^2.} \tag{70}$$

Note that for a perfect conductor, the sign of the electrostatic pressure is always positive, no matter the sign of the electrical field at the surface. Indeed, if the value of the external field is reversed, the sign of the surface charges is reversed too, so that the net electrostatic pressure remains positive, inducing tension.

3.2 Communication between the different computational domains

Before implementing a communication between both solvers, an important step has been for each solver to focus only on its subdomain. For this purpose, the whole two-dimensional domain Ω is split into two domains: Ω_{FEM} and Ω_{BEM}. This is done by creating specific physical surfaces in the `.geo` file passed as an argument to the solver. Consequently, the BEM solver only focuses on the nodes and elements in the BEM domain, while the FEM solver only focuses on the ones in the FEM domain. To allow such a behaviour, a map between the global node/element tag and the corresponding index in the particular subdomain is used.
First, the BEM solver retrieves the electrostatic solution in the whole BEM domain. This step does not require any interaction between both subdomains.
In this first step, the electric field on the elements between both subdomains has been computed. It is precisely the electric field at the boundary between the two subdomains that

induces an electrostatic pressure which acts as a Neumann boundary condition on the FEM domain. To ease the communication between both solvers, a physical curve named *FEM-BEM-Boundary* corresponding to this particular boundary has been defined in the `.geo` file. After the resolution of the electrostatic problem, the BEM solver loops on the elements of this physical curve to fill a map assigning an electric field to each element tag. More precisely, the electric field is directly converted into an electrostatic pressure according to the relation derived previously.
This particular map is then passed to the FEM solver and used to compute nodal forces associated to the nodes of the corresponding element. This is done by assuming the electric field and the electrostatic pressure to be constant along one element (reasonable assumption is the mesh is sufficiently refined) and performing a numerical integration using Gauss quadrature to assign a consistent nodal force to each node of the element. Hence, the electrostatic pressure can be seen as one more Neumann boundary condition for the FEM solver, as it also contributes to the nodal forces vector $\boldsymbol{f}$.
Eventually, the FEM solver computes the solution to the elastic problem in the FEM subdomain.

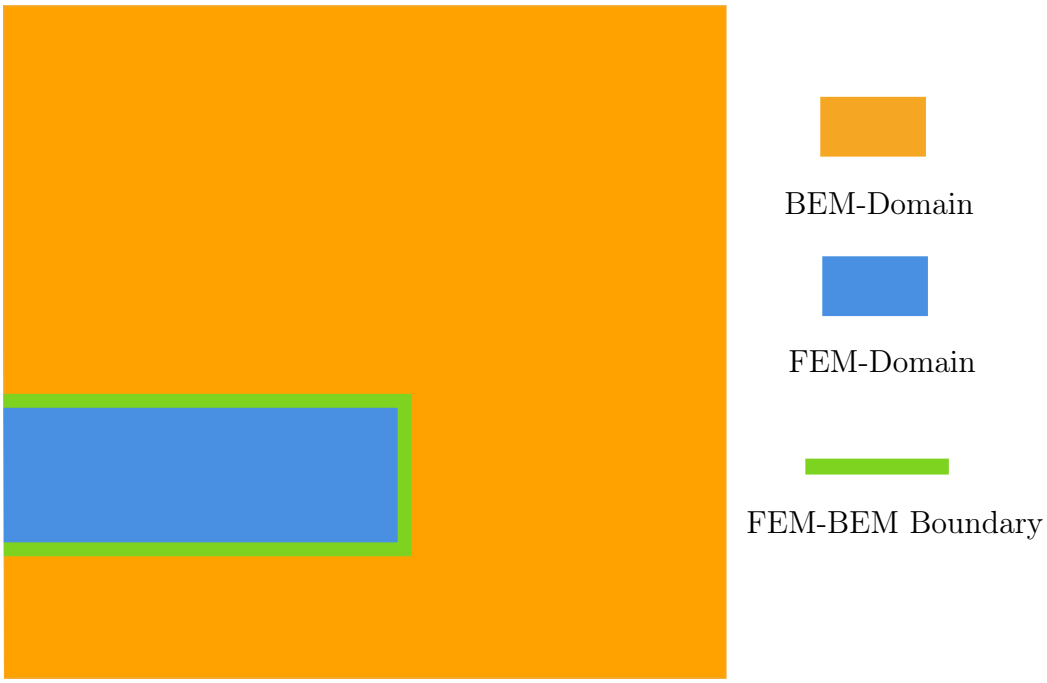


Figure 33: Example of some geometry subdivided into a BEM domain Ω_{BEM} (in orange), a FEM domain Ω_{FEM} (in blue) and their boundary (in green). It allows a simple communication between both solvers.

3.3 Validation

In order to validate the coupling between the BEM solver and the FEM solver, i.e. the communication of the electrostatic pressure from the BEM solver to the FEM solver, the geometry represented in Figure 34 is considered. It is constituted from a solid block (in the FEM domain Ω_{FEM}) and an exterior region at its right which is the BEM domain Ω_{BEM}. In the BEM domain, the prescribed electrostatic boundary conditions should theoretically induce a constant uniform electric field $|E| = \Delta V / L_{\text{BEM}} = 3 \cdot 10^7$ [V/m] pointing to the right.

The uniform field should act on the right edge of the solid block (the boundary between the two subdomains) as a constant surface traction of $p_e = \varepsilon_0 |E|^2/2 = 3982.5$ [Pa], considering the permittivity of the vacuum $\varepsilon_0 = 8.85 \cdot 10^{-12}$ [F/m]. Hence, from a mechanical point of view, it corresponds to a simple tension configuration similar to the one studied in Figure 1.

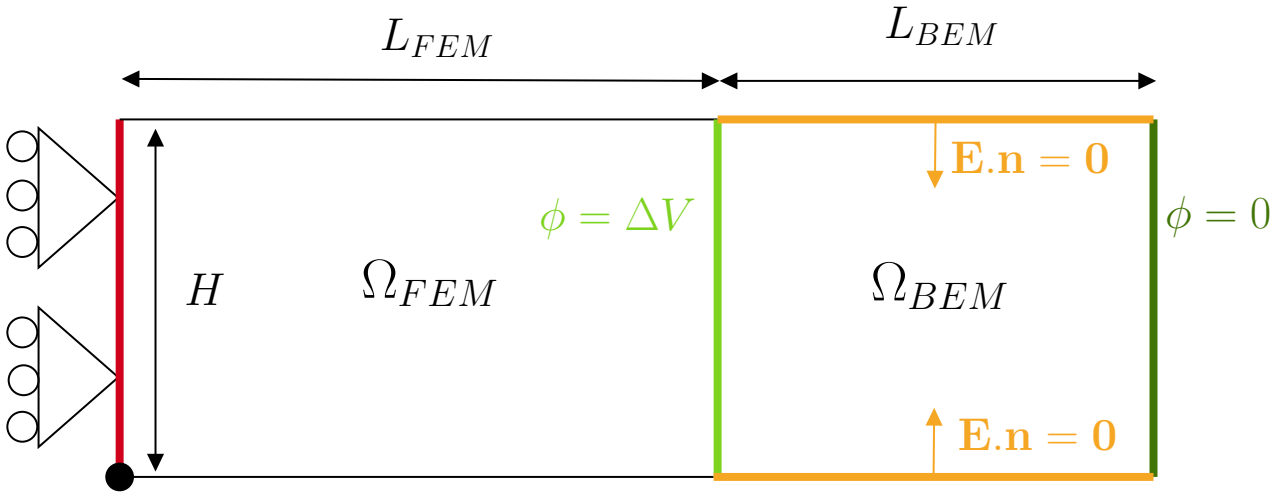
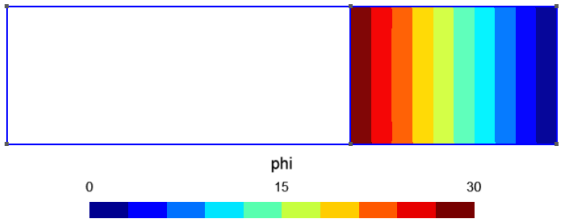


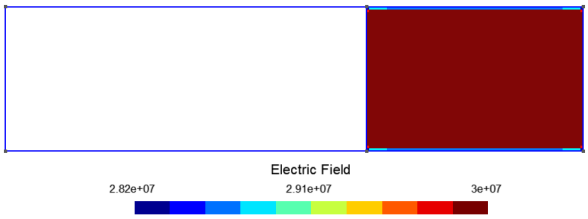
Figure 34: Validation geometry and corresponding boundary conditions for the linear FEM-BEM coupled solver. Note that the dimensions are not at scale. The domain of height $H = 2$ [µm] is split in Ω_{FEM} and Ω_{BEM}. The length of the FEM domain is $L_{\text{FEM}} = 5$ [µm] and the length of the BEM domain is $L_{\text{BEM}} = 3$ [µm]. No horizontal displacement is allowed on the left edge (in red), while the vertical displacement is set to $u_y = 0$ at the bottom left corner. On the middle edge at the boundary of both subdomains (in light green), an electrostatic Dirichlet boundary condition $\phi = \Delta V = 90$ [V] is imposed, while the right edge (in dark green) corresponds to the mass ($\phi = 0$ [V]). Finally, a Neumann boundary condition, ensuring the normal component of the electrical field $\boldsymbol{E} \cdot \boldsymbol{n} = 0$ [V/m] is applied on the remaining boundary of the BEM domain.

This geometry is tested with a regular mesh composed of first order quadrangles, using 50 [-] elements along H, 125 [-] elements along L_{FEM} and 75 [-] elements along L_{BEM}. The corresponding `coupling_validation.geo` file is available on the master branch. The numerical results are represented in Figure 35. As expected, the electric potential involves linearly in the BEM domain, while the electric field is uniform. Note that at the boundary of the BEM domain, the electric field is not exactly equal to its expected value. In the FEM domain, the resulting axial stress is almost uniform and equal to its theoretical value. However, as explained in Section 2.10, the numerical instabilities observed when the BEM solver encounters corners of the domain seem to introduce some imprecision in the numerical value of the electric field, hence leading in some error when passing the electrostatic pressure to the FEM solver. Indeed, at the corners of the BEM/FEM boundary, the axial stress does not correspond exactly to its theoretical value.

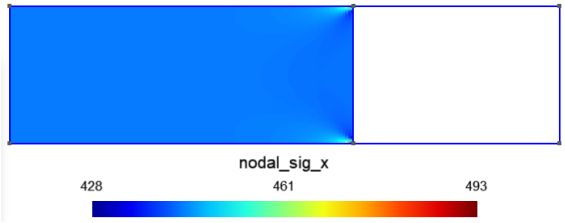
However, this numerical effect is reduced when refining the mesh. As a conclusion, the communication between both solvers works properly.



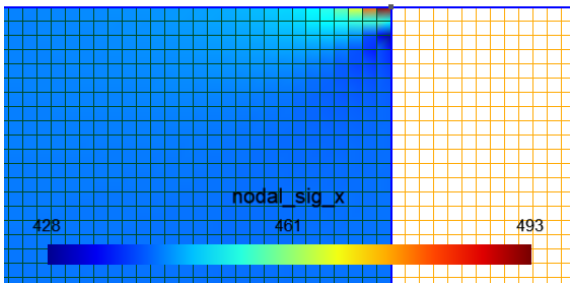
(a) Electric potential field ϕ [V].



(b) Electric field $\|\boldsymbol{E}\|$ [V/m].



(c) Axial stress field σ_{xx} [Pa].



(d) Axial stress field σ_{xx} [Pa] zoomed on the upper right part of the block.

Figure 35: Electric potential field, electric field and axial stress field obtained for the validation geometry described in Figure 34. Note that the mesh is not displayed to improve the readability.

4 Non-linear FEM solver

In order to tackle more realistic applications involving large rotations of the elastic structure, a non-linear geometric finite element method solver, using the *corotational approach*, is implemented. It is an adaptation of the linear FEM solver, in which an angle of rotation is assigned to each finite element, defining the rotation between a global (and fixed) system of axes and the local system of axes attached to the element. Please note that the implemented algorithm allows to deal with large displacements, not with large deformations which would require a more complex theory involving other strain tensors, cfr. [4].
An iterative Newton-Raphson algorithm is implemented to compute displacement increments. Hence, this non-linear solver works with a *tangent* stiffness matrix.
The method presented in this section has been validated in [5] for 4-nodes element. In this project, it has been adapted to triangular and higher order elements.
First, the corototational approach is described before detailing the implementation of the iterative procedure.

4.1 The corotational approach

The corotational approach allows to decompose the total motion into a rigid body motion and a deformational part. The rigid body motion consists of a translation of the center of the element, defining the origin of the local system of axes, and the rotation of the element around its center. In the elemental local system of axes, the deformations are still assumed to be small. Therefore, the linear elasticity theory described previously allows to compute a local elemental stiffness matrix. Using some transformation matrices, it is possible to retrieve the corresponding expression in the global reference system of axes. However, as the rigid body motion can be large, the non-infinitesimal rotation of each finite element allows the global structure to reach large displacements.

4.1.1 The kinematics of one finite element

In this section, a finite element with an arbitrary number of nodes n is considered. The global coordinates (i.e. in the reference system of axes - in the undeformed configuration) of the node i are denoted as (X_i, Y_i), while its global displacements (at a given iteration of the procedure) are denoted as (U_i, V_i). For each element, the coordinates (X_C, Y_C) and the global displacement (U_C, V_C) of the center of the local system axes is computed as:

$$X_C = \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad Y_C = \frac{1}{n}\sum_{i=1}^{n} Y_i, \qquad U_C = \frac{1}{n}\sum_{i=1}^{n} U_i, \qquad V_C = \frac{1}{n}\sum_{i=1}^{n} V_i. \tag{71}$$

Note that the coordinates (X_C, Y_C) can only be computed once for a given element, at the beginning of the iterative algorithm, while the displacement (U_C, V_C) describing the rigid body translation of the element must be updated at each iteration.
The local coordinates (i.e. in the system of axes translated by (U_C, V_C) and rotated by an angle θ with respect to the reference system of axes) of the node i are denoted as $(x_i, y_i) = (X_i - X_C, Y_i - Y_C)$, while its local displacements are denoted as (u_i, v_i) and are computed as:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x_i + U_i - U_C \\ y_i + V_i - V_C \end{pmatrix} - \begin{pmatrix} x_i \\ y_i \end{pmatrix}. \tag{72}$$

The angle of rigid rotation θ is computed in such a way to minimize the euclidian norm of the local displacements

$$\sum_{i=1}^{n} u_i^2 + v_i^2 \tag{73}$$

and can be obtained as:

$$\tan(\theta^*) = \frac{\sum_{i=1}^{n} \left(x_i \left(y_i + V_i - V_C\right) - y_i \left(x_i + U_i - U_C\right)\right)}{\sum_{i=1}^{n} \left(x_i \left(x_i + U_i - U_C\right) + y_i \left(y_i + V_i - V_C\right)\right)}. \tag{74}$$

The angle θ is either θ^* or $\theta^* + \pi$, depending on which one minimizes the euclidian norm of the local displacements.

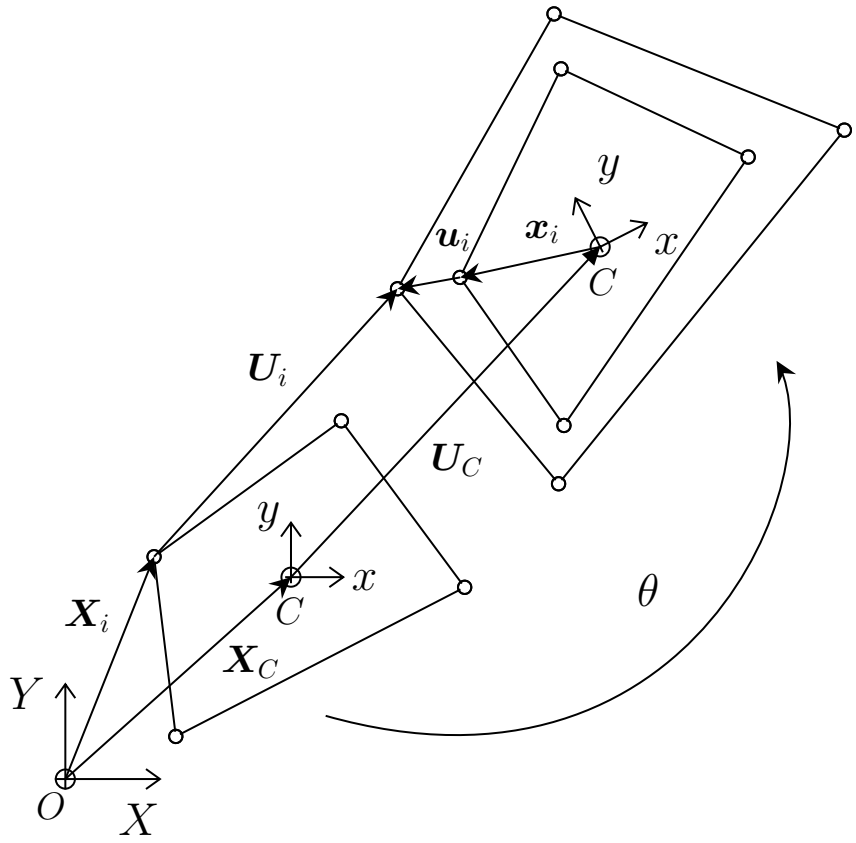


Figure 36: Representation of the different variables associated to the kinematics of one finite element. The global system of axes is denoted (O, X, Y) and the local system of axes is denoted (C, x, y). The global coordinates of the node i is $\boldsymbol{X}_i = (X_i, Y_i)$ and its global displacement is $\boldsymbol{U}_i = (U_i, V_i)$. The coordinates of the center of the element is $\boldsymbol{X}_C = (X_C, Y_C)$ and its displacement is $\boldsymbol{U}_C = (U_C, V_C)$. The local coordinates of the node i is $\boldsymbol{x}_i = (x_i, y_i)$ and its local displacement is $\boldsymbol{u}_i = (ui, v_i)$.

4.1.2 The corresponding transformation matrices

In the theoretical description of the linear FEM solver, an elemental stiffness matrix $\boldsymbol{K}^e$ has been derived. In this section, it corresponds to the local elemental stiffness matrix $\boldsymbol{K}_l^e$, computed in the local system of axes. It can be computed only once for each element at the beginning of the iterative procedure, in the undeformed configuration, in the exact same fashion it is computed in the linear solver. As the local stiffness matrix has been derived in

an energy-consistent way, local nodal forces vector $\boldsymbol{f}_l^e$ can be retrieved from the local nodal displacements $\boldsymbol{d}_l^e$:

$$\boldsymbol{f}_l^e = \boldsymbol{K}_l^e \ \boldsymbol{d}_l^e, \qquad \text{with} \quad \boldsymbol{d}_l^e = \begin{pmatrix} u_1 & v_1 & u_2 & ... & u_n & v_n \end{pmatrix}^T. \tag{75}$$

The correspondance between the global system of axes and the local system of axes of the element is performed introducing one main elemental transformation matrix $\boldsymbol{C}$, which differs from the simple rotation matrix $\boldsymbol{E}$, but ensures that the work done by internal forces is the same in both local and global systems of coordinates (the derivation can be found in [5]). It can be computed as:

$$\boldsymbol{C} = \boldsymbol{P}\boldsymbol{E}^T, \qquad \text{with} \quad \boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}\boldsymbol{G}, \tag{76}$$

$$\boldsymbol{E} = \text{diag}(\underbrace{\boldsymbol{R}, \boldsymbol{R}, ..., \boldsymbol{R}}_{\text{n } \boldsymbol{R} \text{ matrices}}), \qquad \boldsymbol{R} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}, \tag{77}$$

$$\boldsymbol{A} = \begin{pmatrix} -[v_1 + y_1] & [u_1 + x_1] & -[v_2 + y_2] & ... & -[v_n + y_n] & [u_n + x_n] \end{pmatrix}^T, \tag{78}$$

$$\boldsymbol{G} = \frac{1}{\sum_{i=1}^{n}(x_i(u_i + x_i) + y_i(v_i + y_i))} \begin{pmatrix} -y_1 & x_1 & -y_2 & ... & -y_n & x_n \end{pmatrix}. \tag{79}$$

The elemental nodal forces vector in the global axes $\boldsymbol{f}_g^e$ is retrieved as:

$$\boldsymbol{f}_g^e = \boldsymbol{C}^T \ \boldsymbol{f}_l^e. \tag{80}$$

By associating to each node of the element a global node index, the full nodal forces vector $\boldsymbol{f}_g$ can be assembled at each iteration based on each elemental $\boldsymbol{f}_g^e$.
The elemental contribution $\boldsymbol{K}_g^e$ to the full tangent stiffness matrix $\boldsymbol{K}_g$ can be computed as:

$$\boldsymbol{K}_g^e = \boldsymbol{C}^T \ \boldsymbol{K}_l^e \ \boldsymbol{C} + \boldsymbol{K}_h^e, \tag{81}$$

with $\boldsymbol{K}_h^e$ ensuring that the global tangent stiffness matrix involved in the Newton-Raphson algorithm is defined in a consistent way (once again, the details can be found in [5]):

$$\boldsymbol{K}_h^e = \boldsymbol{E}\left[-\boldsymbol{F}^T\boldsymbol{G} - \boldsymbol{G}^T\boldsymbol{F}\boldsymbol{P}\right]\boldsymbol{E}^T, \tag{82}$$

$$\text{with} \quad \boldsymbol{F} = \begin{pmatrix} F_2 & -F_1 & F_4 & ... & F_{2n} & -F_{2n-1} \end{pmatrix}, \tag{83}$$

$$\text{defined such that} \quad \boldsymbol{P}^T\boldsymbol{f}_l^e = \begin{pmatrix} F_1 & F_2 & F_3 & ... & F_{2n-1} & F_{2n} \end{pmatrix}. \tag{84}$$

Once the elemental contribution $\boldsymbol{K}_g^e$ has been computed for each element, the full tangent stiffness matrix $\boldsymbol{K}_g$ is assembled using the global node indexation. It is assembled as a sparse matrix to speed up the computation.

4.2 The iterative procedure

A summary of the iterative algorithm is represented in Figure 37. The details of the procedure are described in this section.

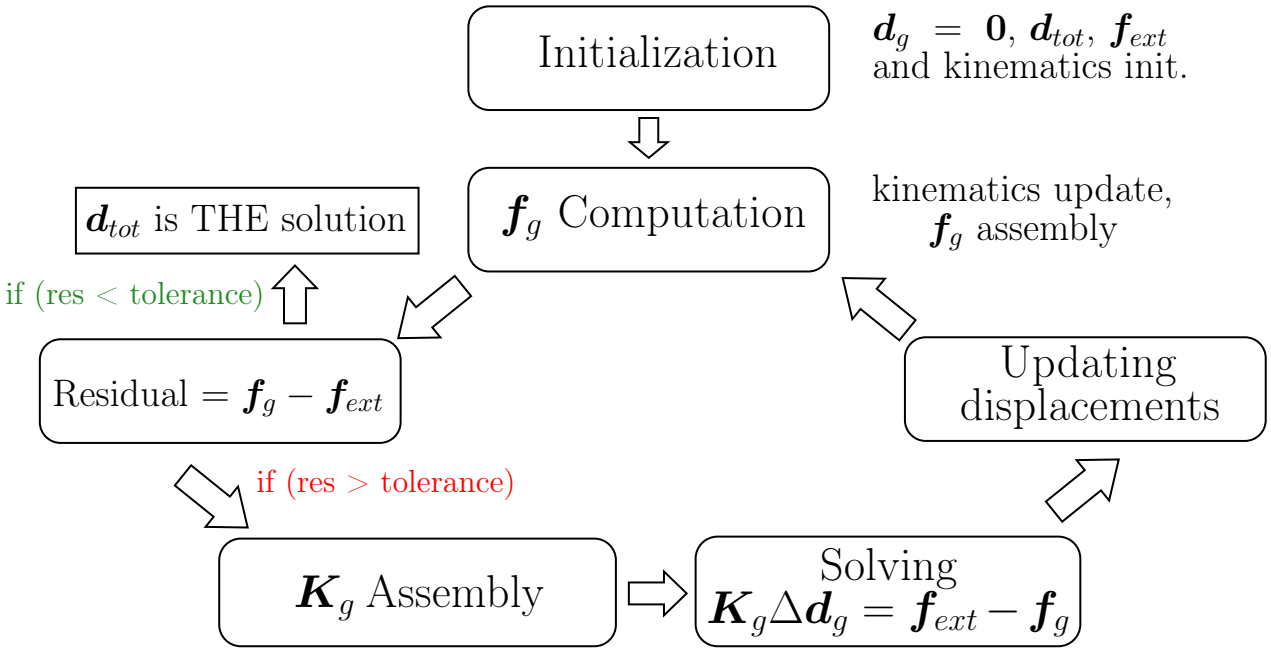


Figure 37: Flowchart representing the iterative procedure implemented in the non-linear FEM solver.

Initialization:

First, the vector $\boldsymbol{d}_{tot}$ containing all the nodal displacements is initialized at 0. If a Dirichlet boundary condition (imposed displacement) is prescribed at a given node, the corresponding value is also inserted in $\boldsymbol{d}_{tot}$. In this case, the value does not vary during the whole iterative procedure, as the unknown global nodal displacement vector $\boldsymbol{d}_g$ contains only the current nodal displacements corresponding to the degrees of freedoms where no Dirichlet boundary condition is imposed. The unknown vector $\boldsymbol{d}_g$ is fully initialized at 0. Moreover, the nodal external forces vector $\boldsymbol{f}_{ext}$, corresponding to the discretization of the Neumann boundary conditions, the volumic forces and the electrostatic pressure (in the case of BEM-FEM coupling), is computed in a similar fashion as in the linear FEM solver. As a reminder, $\boldsymbol{f}_{ext}$ can be decomposed into a volumic contribution of the volumic forces and a surfacic contribution of the prescribed surface tractions (both purely mechanical Neumann boundary conditions and electrostatic pressure application). The numerical integration of the different terms is performed using Gauss quadrature integration rule. Afterwards, the $\boldsymbol{f}_{ext}$ vector is reduced to the degrees of freedom corresponding for which there is not any prescribed Dirichlet boundary condition.

The different variables associated to the kinematics of each element are initialized and stored in an `elementData` data structure gathering every vector and matrix introduced in the previous section. The n node tags, the global (and local !) coordinates of the nodes, as well as the coordinates of the center of the element are computed once and they do not require any update during the iterative procedure. The local and global displacements are initialized at zero, as well as the rigid rotation angle θ. The $\boldsymbol{R}$, $\boldsymbol{E}$, $\boldsymbol{A}$, $\boldsymbol{G}$, $\boldsymbol{P}$ and $\boldsymbol{C}$ matrices are initialized according to their respective definitions.

Finally, for each element, the elemental stiffness matrix $\boldsymbol{K}_l^e$ in the local axes is computed as described for the linear solver. It is also stored in the same data structure as the kinematics-related variables.

Moreover, a vector containing the different 2D element tags in the FEM domain and a `map`

assigning the corresponding `elementData` structure to each element tag are initialized.

Global internal forces computation:
This second step is the beginning of the iterative procedure. It consists in filling the full global nodal forces vector $\boldsymbol{f}_g$. It gathers the nodal forces associated to the free degrees of freedom (all the degrees of freedom except the ones for which there is a prescribed Dirichlet boundary condition), similarly to the unknown global displacement vector $\boldsymbol{d}_g$.
After the initialization (at the first iteration), the local nodal displacements $\boldsymbol{d}_l^e$ associated to each element are equal to zero, so are the local nodal forces $\boldsymbol{f}_l^e$ and the global elemental nodal forces $\boldsymbol{f}_g^e$.
First, for each element, the global displacements (U_i, V_i) of all n nodes are updated according to the complete nodal displacement vector $\boldsymbol{d}_{tot}$ (gathering $\boldsymbol{d}_g$ and the Dirichlet boundary conditions) obtained at the end of last iteration. Note that after the initialization, $\boldsymbol{d}_{tot}$ only contains the prescribed Dirichlet boundary conditions. Several kinematics variables are updated according to the nodal displacements, such as the displacement (U_C, V_C) of the center of the element, the angle of rigid body rotation θ, the local displacements (u_i, v_i) of each node which form $\boldsymbol{d}_l^e$, as well as the complete set of transformation matrices $\boldsymbol{A}$, $\boldsymbol{G}$, $\boldsymbol{R}$, $\boldsymbol{E}$, $\boldsymbol{P}$ and $\boldsymbol{C}$.
For each element, the local nodal forces are retrieved as $\boldsymbol{f}_l^e = \boldsymbol{K}_l^e \ \boldsymbol{d}_l^e$ and converted into global elemental forces as $\boldsymbol{f}_g^e = \boldsymbol{C}^T \ \boldsymbol{f}_l^e$. Finally, using a mapping between the node tags and the indices of each unknown nodal displacement, the full nodal forces vector $\boldsymbol{f}_g$ is assembled. Note that at the first iteration, it only contains the nodal internal forces resulting from the initial deformation corresponding to the prescribed Dirichlet boundary conditions.

Residual computation and stopping criterion:
The residual of the global nodal forces are computed as the difference between the internal nodal forces $\boldsymbol{f}_g$, resulting from each elemental local displacements, and the external nodal forces $\boldsymbol{f}_{ext}$ corresponding to the applied external forces. Ideally, the internal forces should be equal to the internal forces. However, as the change of geometry induces non-linearities, the residual is never exactly equal to zero and some tolerance must be introduced.
In order for the code to be able to tackle problems in which only displacements are prescribed (in such a case, $\boldsymbol{f}_{ext} = 0$ could lead to an inexact solution), it has been decided to focus on the relative difference between two successive *full* nodal forces vectors as a stopping criterion. More precisely, denoting as $f_i^{(n)}$ the nodal force associated to the degree of freedom indexed by i at the iteration n, the norm ϵ of the relative nodal force difference at the iteration n is computed as:

$$\epsilon = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\left|f_i^{(n)} - f_i^{(n-1)}\right|}{\max_{1\leq j\leq N}\left|f_j^{(n)}\right|}\right)^2}, \tag{85}$$

in which $N = 2m$ denotes the number of degrees of freedom associated to a mesh constituted of m nodes.
If ϵ is lower than a fixed tolerance of 10^{-10} determined empirically, the full nodal displacements $\boldsymbol{d}_{tot}$ obtained at the end of last iteration correspond to the solution of the problem. Otherwise, the iterative procedure has not converged yet and further nodal displacement increments must be computed. Note that such a strict tolerance can be chosen as the implemented Newton-Raphson algorithm converges rapidly, as discussed in Section 4.6. In the case such a strict tolerance can not be reached, the maximal number of iterations has been set to 200. Moreover, as shown in Section 4.6, it is possible that the relative nodal force difference

converges towards a value which is greater than the fixed tolerance. A mechanism has been implemented in order to avoid computing tens of iterations without improving the accuracy of the numerical solution. As the convergence is mostly monotonic (the nodal force difference does decrease at each iteration, except sometimes at the second iteration), the number of times the nodal force difference increases between two iterations is count. When such a phenomenon has occurred five times, convergence is assumed and the program is stopped. To take the purely divergent cases into account, the final residual is displayed on screen so that the user can check if its value is low (a value of around $\epsilon \approx 10^{-8}$ is already acceptable).

Global tangent stiffness matrix assembly:
To compute the next nodal displacement increment $\Delta \boldsymbol{d}_g$, the global tangent stiffness matrix $\boldsymbol{K}_g$ must first be computed.
For each element, as the different transformation matrices have already been updated and the local nodal forces have already been computed at the current iteration, the $\boldsymbol{K}_h^e$ matrix ensuring the consistency of the procedure can be computed according to its definition. Similarly, the elemental contribution $\boldsymbol{K}_g^e$ to the global tangent stiffness matrix is also retrieved according to its definition.
Once again, using the mapping between the node tags and the indices of the free degrees of freedom, the global tangent stiffness matrix $\boldsymbol{K}_g$ is assembled as a sparse matrix, using a pre-allocated `Triplet` vector, similarly to what is done in the linear solver.

Computing the nodal displacement increments:
The incremental nodal displacements vector $\boldsymbol{d}_g$ is simply obtained solving the linear system:

$$\boldsymbol{K}_g \ \Delta \boldsymbol{d}_g = \boldsymbol{f}_{ext} - \boldsymbol{f}_g. \tag{86}$$

The right hand side term has already been computed previously as it is the opposite of the residual of the nodal forces. As the tangent stiffness matrix $\boldsymbol{K}_g$ is positive definite and symmetric, the linear system can be solved efficiently using a `SimplicialLDLT` object from the `Eigen` library (similar to the solving procedure of the single linear system of the linear FEM solver).

Updating the global nodal displacements:
The vector of unknown global nodal displacements is updated as

$$\boldsymbol{d}_g^{(i+1)} = \boldsymbol{d}_g^{(i)} + \Delta \boldsymbol{d}_g^{(i)}, \tag{87}$$

with $\boldsymbol{d}_g^{(i)}$ denoting the value of the vector at the beginning of the i-th iteration of the procedure and $\Delta \boldsymbol{d}_g^{(i)}$ the corresponding increment.
Moreover, the full vector of nodal displacements $\boldsymbol{d}_{tot}$, gathering the displacements of every node (even the ones which are constrained with Dirichlet boundary conditions), is also updated.

Finally, the iterative procedure keeps going as the global internal forces $\boldsymbol{f}_g$ are computed according to the updated nodal displacements (see step *Global internal forces computation*).

4.2.1 Parallelization

The code is parallelized using `openmp`. The kinematics update (first step of the iterative procedure) consists of one loop over the FEM element tags, it is parellelized using one single

`#pragma omp parallel for` command. The global nodal forces $\boldsymbol{f}_g$ assembly can be parallelized similarly, as local elemental vectors must be computed before writing the global corresponding results in the global nodal forces vector. To avoid data race between different threads, `#pragma omp atomic update` commands are used. For the assembly of the global tangent stiffness matrix $\boldsymbol{K}_g$, a vector of triplets is initialized for each thread, avoiding data race once again. Afterwards, the different vectors are assembled in one single vector of triplets. To speed up the code, the vectors are initialized using the `.reserve(approx_size)` method, with `approx_size` an estimation of the final size of the vector computed by making the assumption each element inserts 64 values in the sparse global matrix. This estimation is accurate in the case of first order quadrangles, otherwise it provides a first order of magnitude which is reasonable.

4.3 Post-processing

First, to ease the iterative coupling between the FEM solver and the BEM solver, the displacements of the nodes located on the FEM-BEM boundary are retrieved as pairs and stored into a map assigning to each node tag its nodal displacements. The utility of such a data structure is detailed in the non-linear iterative FEM-BEM solver section.
Once the final nodal displacements $\boldsymbol{d}_{tot}$ are known, the final nodal forces (at each node, even at the nodes for which there is some prescribed Dirichlet boundary condition) can be retrieved using an assembly procedure similar to the one described in the *Global internal forces computation* step. From the local elemental nodal displacements $\boldsymbol{d}_l^e$, the local elemental nodal forces are retrieved as $\boldsymbol{f}_l^e = \boldsymbol{K}_l^e\ \boldsymbol{d}_l^e$ and converted into global axes using $\boldsymbol{f}_g^e = \boldsymbol{C}^T\ \boldsymbol{f}_l^e$. Finally, the complete nodal forces vector can be assembled by assigning a global index to each node of the FEM domain.
Once the nodal forces are known, the reaction forces can be computed on the surfaces at which the displacement is prescribed by Dirichlet boundary conditions, by summing the nodal forces of the nodes belonging to the given surface.
The final step is to recover the different strains and stresses inside each element. As described in the linear FEM solver, the strains inside one element can be recovered using following relation:

$$\begin{bmatrix} \varepsilon_{xx}^l \\ \varepsilon_{yy}^l \\ \gamma_{xy}^l \end{bmatrix} = \boldsymbol{B}\boldsymbol{d}_l^e, \qquad \text{with} \quad \boldsymbol{B} = \begin{bmatrix} \partial_x & 0 \\ 0 & \partial_y \\ \partial_y & \partial_x \end{bmatrix} \begin{bmatrix} N^1 & 0 & N^2 & .. \\ 0 & N^1 & 0 & .. \end{bmatrix}. \tag{88}$$

Because the reduced system of coordinates (ξ, η) corresponding to one element is attached to the particular element, the exact same method as the one described in the linear solver can be used to compute the strain-displacement $\boldsymbol{B}$ matrix.
Moreover, as Hooke's law of elasticity is isotropic, the Hooke's matrix $\boldsymbol{H}$ introduced in the linear FEM solver allows to retrieve the stresses expressed in the local axes:

$$\begin{bmatrix} \sigma_{xx}^l \\ \sigma_{yy}^l \\ \sigma_{xy}^l \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} \varepsilon_{xx}^l \\ \varepsilon_{yy}^l \\ \gamma_{xy}^l \end{bmatrix}. \tag{89}$$

As explained previously, the local strains and stresses can be evaluated directly at the nodes (and not at the Gauss points) of each element as linear elasticity is considered locally.
To retrieve the strains and stresses expressed in the global axes, one must first re-construct the corresponding tensors in the local axes as:

$$\boldsymbol{\varepsilon}_l = \begin{bmatrix} \varepsilon_{xx}^l & \varepsilon_{xy}^l \\ \varepsilon_{xy}^l & \varepsilon_{yy}^l \end{bmatrix}, \qquad \boldsymbol{\sigma}_l = \begin{bmatrix} \sigma_{xx}^l & \sigma_{xy}^l \\ \sigma_{xy}^l & \sigma_{yy}^l \end{bmatrix}, \tag{90}$$

in which the shear strain is given by $\varepsilon^l_{xy} = \gamma^l_{xy}/2$. Knowing that for one element, the relation between $\boldsymbol{e}_X$ and $\boldsymbol{e}_Y$, the unit basis vectors of the global system of coordinates (O, X, Y), and $\boldsymbol{e}_x$ and $\boldsymbol{e}_y$, the unit basis vectors of the local system of coordinates (C, x, y), can be expressed as:

$$\boldsymbol{e}_X = \cos(\theta)\ \boldsymbol{e}_x - \sin(\theta)\ \boldsymbol{e}_y, \qquad \boldsymbol{e}_Y = \sin(\theta)\ \boldsymbol{e}_x + \cos(\theta)\ \boldsymbol{e}_y, \tag{91}$$

with θ the angle of rotation of the particular element, it is finally possible to obtain the expression of the tensors in the global axes using the rotation matrix $\boldsymbol{R}$ introduced in Equation 77,

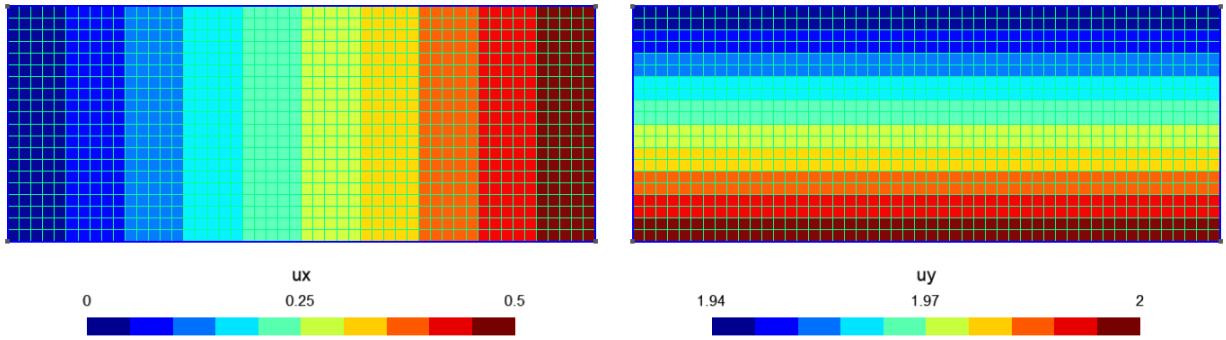
$$\boldsymbol{\varepsilon}_g = \boldsymbol{R}\ \boldsymbol{\varepsilon}_l\ \boldsymbol{R}^T, \qquad \boldsymbol{\sigma}_g = \boldsymbol{R}\ \boldsymbol{\sigma}_l\ \boldsymbol{R}^T. \tag{92}$$

Moreover, the total potential energy TPE can be computed in a similar fashion as introduced by Equation 16. For the numerical integration of the internal strain energy U, there is no need to convert the elemental strain and stress tensors back to global system of axis, the internal strain energy being a scalar, it is independent from the system of coordinates.
Please note that in the present non-linear case, Clapeyron's theorem does not hold anymore and $U \neq P/2$.

4.4 Validation

4.4.1 Small displacements configuration

First, the non-linear solver is tested in the exact same geometry as described in Section 1.2 and Figure. 1.
To activate the non-linear solver, please add `SetNumber{"Non_linear_solver",1};` in the `simple_tension.geo` file. As can be observed in Figure 38, the numerical results are similar to the ones obtained with the linear solver. Note that the small displacement assumption is not strictly satisfied as a vertical displacement of around 2 [m] is imposed in order to highlight the fact that the solver is able handle rigid body modes properly. If the superimposed rigid body translation is neglected, the displacements of the structure are indeed small.

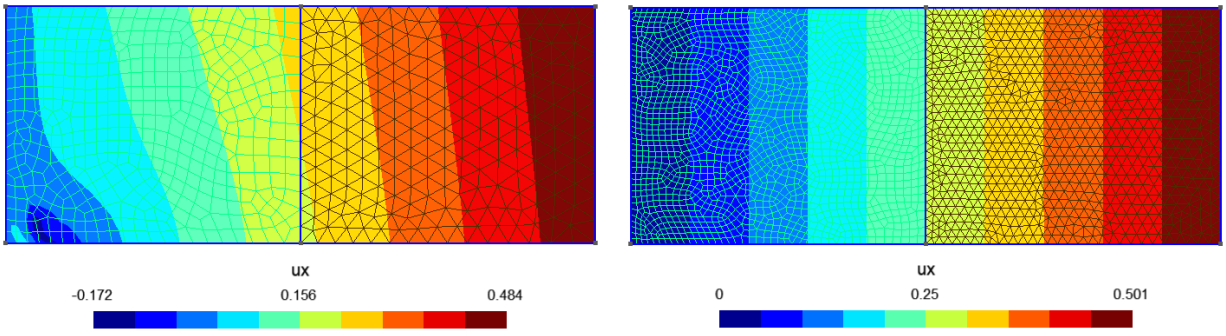


(a) Horizontal displacement field u_x [m]. (b) Vertical displacement field u_y [m].

Figure 38: Displacement field obtained with the non-linear solver in the configuration described by Figure 1, using a mesh of $n_x = 50$ [-] elements along the x-axis and $n_y = 20$ [-] along the y-axis, resulting in square elements of side 0.1 [m].

The non-linear solver also works with an hybrid mesh and higher order elements. However, as can be observed in Figure 39, the prescribed vertical displacement $\bar{u}_y = 2$ [m] of the bottom left node can induce a divergence of the solution if the prescribed displacement is too important (such a divergence has not been observed for $\bar{u}_y = 0.5$ [m]). Strictly speaking,

the solution does not diverge but it does never converge towards the exact solution. This problem can be overcome by increasing the number of finite elements.



(a) (b)

Figure 39: Horizontal displacement field u_x [m] obtained with an hybrid mesh generated with the `complex_validation.geo` file and elements of the second order. In (b), the element density has been increased by a factor of two with respect to the original `complex_validation.geo` file (a).

4.4.2 Angle frame configuration

In this section, the non-linear FEM solver is validated against a geometry which has been studied in [5]. It consists of an angle frame clamped at its left edge with a surface traction applied on its upper edge, as described in Figure 40.

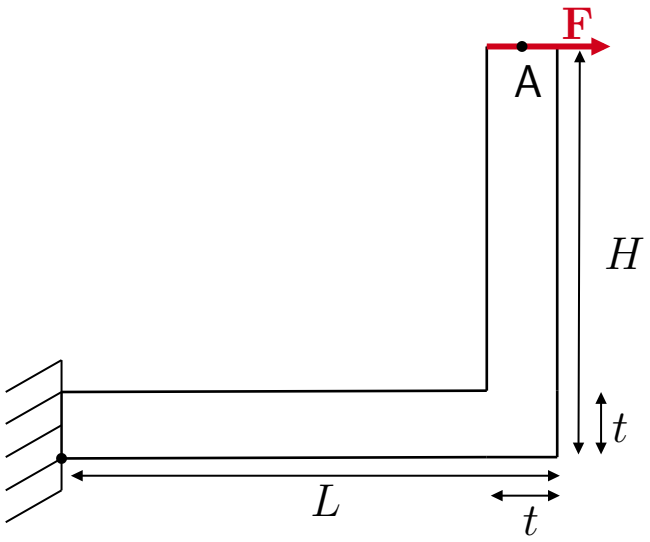
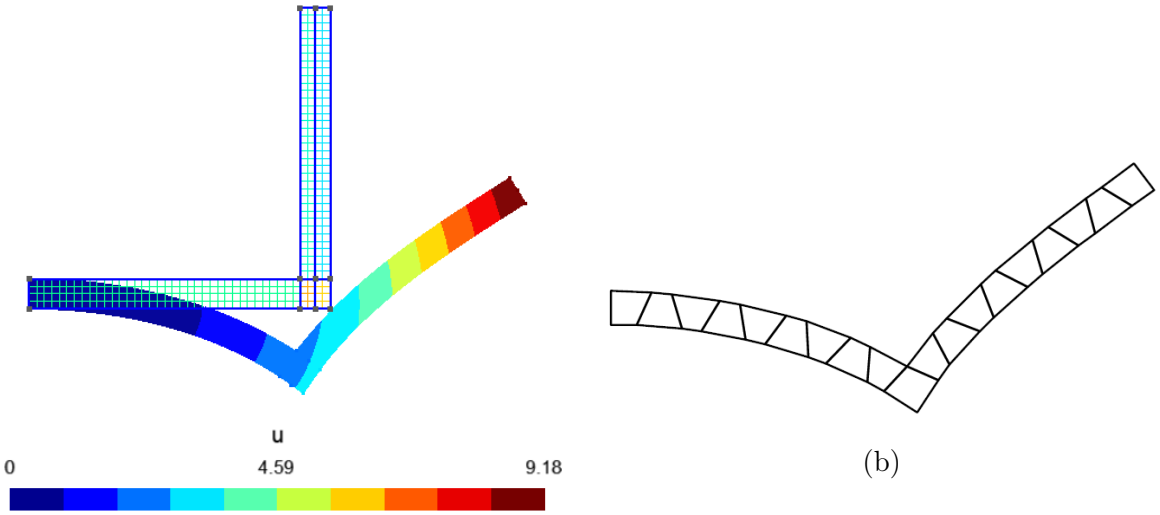


Figure 40: Geometry associated to an angle frame beam. The beam is clamped on its left edge corresponding to an imposed zero displacement $u_x = u_y = 0$. Its Young's modulus is $E = 3 \cdot 10^7$ [Pa] and its Poisson ratio $\nu = 0.3$ [-]. Its total length is fixed to $L = 10$ [m] and its total height to $H = 10$ [m]. The thickness of the beam is constant $t = 1$ [m]. The beam is uniformly pulled to the right at his top edge by an uniform surface traction. The point at the middle of the top edge is denoted point A.

For a total horizontal force $F = 40000$ [N/m], Battini *et al.* [5] obtained an horizontal displacement of the point A of approximately $u_A = 6.75$ [m], using a fine mesh constituted

of 304 square elements. They obtained this result using two different local finite element formulations (two different ways to compute $\boldsymbol{K}_l^e$ in Equation 81). For the same configuration, the results obtained with our implementation are represented in Figure 41a and the horizontal displacement of point A is $u_A = 6.73$ [m]. As can be observed, they are similar to the numerical results obtained by Battini *et al.* using a coarse mesh displayed in Figure 41b. As a conclusion, the corotational formulation has been implemented successfully.
Moreover, as can be seen in Figure 41a, the FEM domain is consisting of five smaller subdomains, which highlights the ability of the code to handle different subdomains.



(a) (b)

Figure 41: (a) Deformed configuration and total displacement $\|\boldsymbol{u}\|$ [m] represented on the initial mesh, obtained with a uniform mesh constituted of 304 square elements of edge 0.25 [m]. (b) Deformation computed by Battini *et al.* [5] using a coarse mesh. This subfigure is directly taken from [5].

n_{nodes} [-]	$n_{\text{iterations}}$ [-]	u_A [m]	CPU time/$n_{\text{iterations}}$ [s]
385	16	6.728	0.0034
1377	13	6.779	0.010
2977	18	6.791	0.018
5185	17	6.796	0.033
11425	14	6.799	0.082
20097	15	6.801	0.149
44737	15	6.802	0.431
79105	13	6.803	1.04
177025	15	6.804	2.50
313857	19	6.804	5.30

Table 4: Evolution of the horizontal displacement u_A [m] at the tip of the angle frame described in Figure 40 as a function of the number of nodes n_{nodes} used to discretize the geometry. The total number of iterations $n_{\text{iterations}}$ and the CPU time per iteration are also represented.

Moreover, one can study the convergence of the horizontal displacement of point A u_A when

refining the mesh. The results are gathered in Table 4.

As can be observed, when using 304 elements (385 nodes) for the computation, the error on u_A with respect to the converged solution is only of 1.12 [%], which shows that the numerical implementation is robust even when dealing with relatively coarse meshes. Moreover, one can observe that the number of iterations does reasonably vary when the number of elements is modified. In most cases, it reaches a final value of the relative nodal force difference, greater than the fixed tolerance, after five to ten iterations, before oscillating slightly and once the nodal force difference has increased five times between two iterations, the program stops automatically.
The von Mises stress field is represented in Figure 42. Note that the scale axis has been modified as the stress at the inner corner ($\sigma_{\text{VM}} = 8.57$ [MPa]) is much greater than in the rest of the structure. The stress field seems to accurately represent the physics, as it corresponds to flexion in the horizontal bar and in the bottom part of the vertical bar, while the tip of the structure is almost stress free.

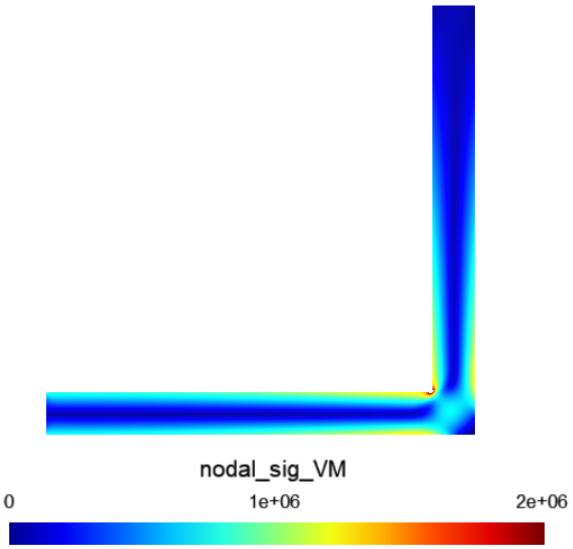


Figure 42: Von Mises stress σ_{VM} [Pa] field for the angle frame geometry described in Figure 40, it has been obtained using 175104 first order square elements, corresponding to 177025 nodes.

4.5 Code performance

To understand why the CPU time per iteration increases with the number of nodes as observed in Table 4, one can study the relative amount of time each part of one iteration takes. The study is performed for the angle frame configuration described previously, in the configuration with 175104 first order square elements and 177025 nodes, using 8 virtual threads (four distinct threads). In average, one iteration of the algorithm takes 2.50 [s]. Updating the kinematics takes 0.08 [s], retrieving and assembling the global nodal forces vector 0.08 [s] (including residual and nodal relative difference computation), assembling the global tangent stiffness matrix 0.34 [s], solving the linear system using the `SimplicialLDLT` object 1.78 [s]. The remaining time (around 0.2 [s]) is spent in the initialization and in smaller parts of the algorithm.
Hence, the computation time and the complexity of the implementation are mainly driven by solving the linear system. Indeed, as can be observed in Figure 43, when increasing the number of degrees of freedom (twice the number of nodes), solving the linear system takes

more and more time compared to the global time of the whole iteration. It indicates that the complexity of our implementation is at most equal to the complexity of the `SimplicialLDLT` algorithm, which performs a direct LDLT Cholesky factorization optimized for sparse positive semi-definite matrices.

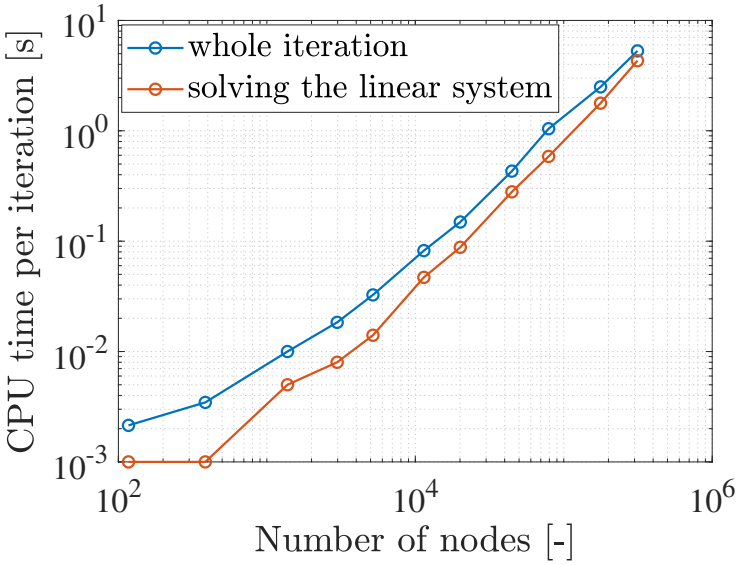


Figure 43: Evolution of the CPU time [s] per iteration and the time [s] for solving the linear system as a function of the number of nodes, in logarithmic scale. The results are measured using the angle frame geometry (`large_rotation_validation.geo`)in the same configurations as described in Table 4.

Performing a linear regression to estimate the slope of the logarithmic curve for the CPU time required to solve the linear system, the complexity of our implementation seems to evolve as

$$\text{CPU Time} \propto n_{\text{nodes}}^{C}, \quad \text{with} \quad C \in [1.35, 1.40] \text{ [s]}. \tag{93}$$

As the most time-consuming part of the algorithm is the resolution of the linear system, it is relevant to compare the performance of the `SimplicialLDLT` solver with other direct solvers of the `Eigen` library. As discussed previously, iterative solvers for linear systems are not considered as they are more likely to diverge and as they induce some inaccuracy.
The comparison is performed using once again the angle frame configuration with 177025 nodes and 8 virtual threads. Table 5 gathers the performance of the principal direct solvers for sparse linear systems available in the `Eigen` library.[3] As can be seen, both `SimplicialLLT` and `SimplicialLDLT` solvers are optimized for positive definite matrices and their performance is comparable, the `SimplicialLLT` is however a bit solver and less stable: divergence has been observed when using it to solve the linear systems for the configuration described by Figure 39(b). The LU decomposition takes more time as it is a more general method. Finally, the QR decompisition can not even be tested for a system of this size. In the configuration with 1377 nodes, it takes 2.38 [s] to solve the linear system once.
As a conclusion, the `SimplicialLDLT` algorithm is the most adapted direct solver for solving the sparse linear systems involved in the FEM computation.

[3]see Eigen documentation for solving sparse linear systems, consulted on May 13, 2022.

	SimplicialLLT	SimplicialLDLT	SparseLU	SparseQR
CPU time [s]	1.855	1.788	5.029	-

Table 5: Average CPU Time [s] required for the different direct solvers for linear systems available in the `Eigen` library to solve the sparse system described by Equation 86 once, for the `large_rotation_validation.geo` file using 177025 nodes. The average is performed on 20 different time measurements.

4.6 Convergence study of the iterative algorithm

The convergence of the residual defined by Equation 85 is studied for various geometries and the results are represented in Figure 44.[4]

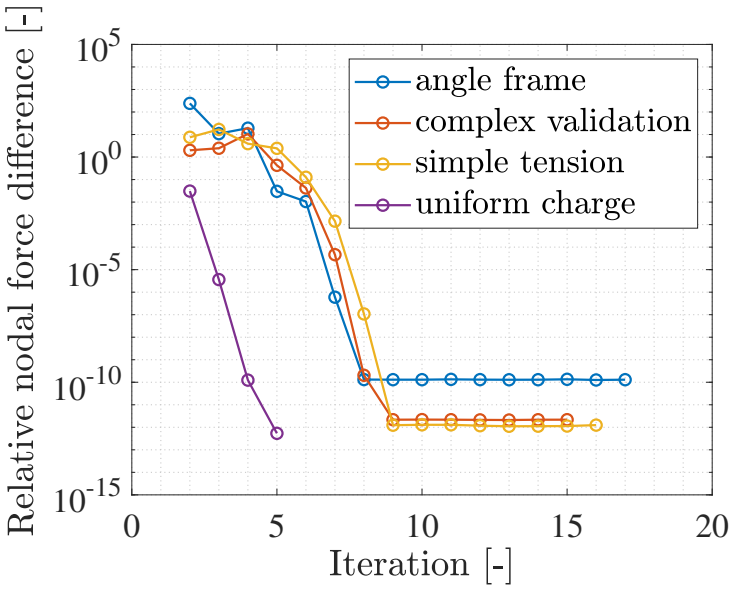


Figure 44: Evolution of the residual of the relative forces defined by Equation 85 of the iterative non-linear FEM solver as a function of the number of iterations for various geometries corresponding respectively to the `large_rotation_validation.geo`, `complex_validation.geo`, `simple_tension.geo` and `uniform_charge.geo` files.

In some configurations, the convergence of the Newton-Raphson algorithm is not motonic during the first iterations. This has been observed in particular for the geometries involving large displacements or non-homogeneous prescribed Dirichlet boundary conditions. However, one common characteristic to the different geometries is the very rapid convergence of the residual, once it has started to decrease monotonically. In this case, the fixed tolerance (or the most precise accuracy which is possible to reach) is reached in a few iterations with a convergence of almost the fourth order, meaning the residual is decreased by a factor 10^{-4} at each iteration. Such a behaviour further validates the correct implementation of the described iterative algorithm.

[4]To reproduce the results, please make sure the non linear solver parameter is activated before running the solver.

5 Non-linear iterative FEM-BEM coupling

For a non-iterative solver, the BEM solver computes the electrostatic pressures that are given to the FEM solver which then computes the nodal displacements of the FEM-BEM boundary nodes (schematized in Figure 33). However, after this structural displacement the electric potential cannot be considered as being the same as before since the geometry of the problem has changed a lot (the goal of the non-linear FEM solver is to deal with large displacements). One should thus solve again the electrostatic problem in order to recompute the electrostatic pressures and so on. Below the pull-in voltage, the deformed body eventually stabilises in an equilibrium configuration.

The goal of the iterative coupled solver is then to make the BEM and the FEM solvers communicate multiple times to exchange respectively the electrostatic pressures and the nodal displacements.

The difference with the one-way coupled solver is that the BEM solver will have to take into account the nodal displacements of the FEM-BEM boundary in order to recompute the electrostatic pressures. Also, the iterative solver will need a condition for stopping the iterations between the two solvers when the nodal displacements of the deformed body have converged.

5.1 Method

The nodal displacements (along x and y) of the FEM-BEM boundary (computed with the FEM solver) are stored under the form of a map that associates a 2D-displacement to each node tag. This map is passed as an argument to the BEM solver. Every time the BEM solver is called, it retrieves everything from the `.geo` file, including the node coordinates. Then it associates to each boundary element the coordinates of its nodes.

What changes here is that for each node belonging to the FEM-BEM boundary, its coordinates are incremented by its displacement (computed by the FEM solver and passed as an argument to the BEM solver) given by the boundary displacement map before being stored in the element structure.

By doing so, the `.geo` file only gives the undeformed geometry, but all the displacements are stored and passed to the BEM solver without the help of any `.geo` file. Sometimes, it can be interesting to visualize the electrical quantities on the deformed mesh. In order to do so, a specific tool (the *mesh untangler*) is used in order to rebuild the mesh in the BEM domain from the displacements of the FEM-BEM boundary. However, the general post-processing of both the FEM and BEM solvers is performed only once when convergence is reached. Indeed, the electric potential and the electric field inside the BEM domain, as well as the strain and stress fields inside the FEM domain, are not required for performing the iterative algorithm. By doing so, the computation time of the whole algorithm is reduced and its performance is increased.

5.2 Stopping criterion

The iterations will continue until the stopping condition is encountered. It should take into account the displacements from the previous and the current iteration and be true when their difference is globally sufficiently small. However, it is better to use a relative difference rather than an absolute difference (between the displacements of two successive iterations)

since the characteristic size of the considered problem could vary a lot. One thus have the relative displacement difference of each node and for each iteration

$$x_i^{(k)} = \frac{\|\boldsymbol{u}_i^{(k)}\| - \|\boldsymbol{u}_i^{(k-1)}\|}{\|\boldsymbol{u}_i^{(k-1)}\|}, \tag{94}$$

with i being the tag of the considered node, k the current iteration, $\boldsymbol{u}$ the vectorial displacement and $\|.\|$ representing the Euclidian norm of a vector.

Also, one should define a norm for transforming the relative displacement difference of each FEM-BEM boundary node into a scalar (some kind of average). The norm that as been adopted here is

$$|\boldsymbol{x}| = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}, \tag{95}$$

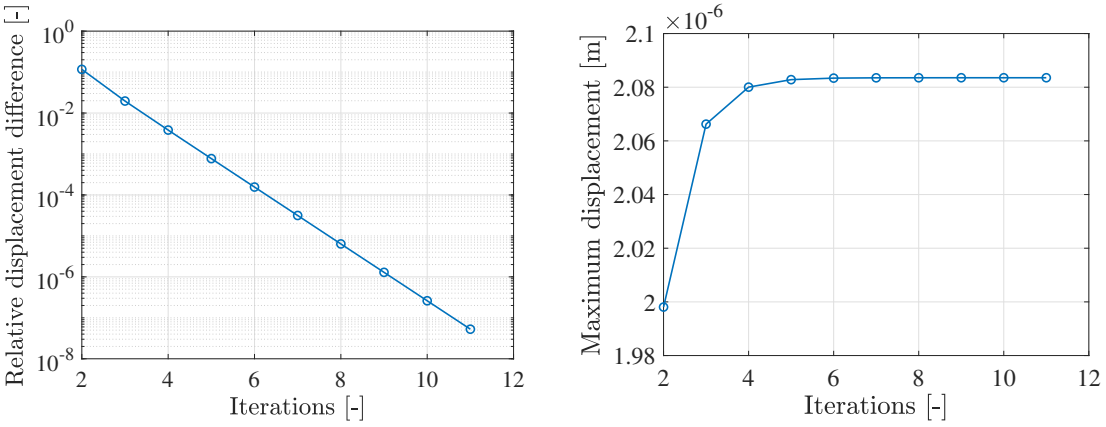
where $|.|$ represents the norm and $\boldsymbol{x}$ is a vector of size N.

The stopping condition is thus, $|\boldsymbol{x}^{(k)}| < \varepsilon$, with ε a user defined threshold, set to 10^{-7} by default.

However, it is possible that the deformed body does not stabilises around an equilibrium point and for this reason, one should add a maximum number of iterations. In fact, in the where the number of iteration reaches the maximum number of iteration, the user should be warned that the solution has diverged. This limit has typically been set around 50 iterations, but for some specific applications (typically pull-in evaluation), this number can be set to a higher value of the order of 100 iterations. In fact, at the edge of pull in, for example, a very high number of iterations is needed for the solver to converge.

5.3 Convergence

It is possible to study the convergence of the solution as a function of the iterations (between the BEM and the FEM solvers). This is what has been done in Fig. 45a and 45b. In this case, the ε parameter has been fixed to 10^{-5} [%] and so iterations where performed until the relative displacement difference was lower than 10^{-5} [%].



(a) Relative displacement difference.

(b) Maximum displacement.

Figure 45: Study of the convergence of the iterative coupled solver as a function of the number of iterations on a geometric configuration described in the `MEMS.geo` file available on the master branch.

It can be seen in Figure 45a that the semi-log plot behaves as a straight line, meaning that the relative displacement difference of the coupled iterative solver exponentially decays with the number of iterations. In order to be more confident with the error, the maximum displacement at each iteration is presented in Figure 45b. This shows that the maximum nodal displacement converges towards a fixed value (even after 5-6 iterations) and that the structure well rapidly converges towards the deformed configuration.

6 Applications

In this section, several geometries and practical MEMS applications are studied using the FEM-BEM solver described in previous sections. The general physical behaviour of the structures, as their displacement-voltage curve, is determined, as well as the stress field and the electrostatic fields.

6.1 Electrostatically actuated micro-beam clamped on one side

The first application corresponds to the electrostatic actuation of a clamped micro-beam made of silicon. Its basic geometry allows to fully understand the physics related to the electrostatic actuation of microsystems, such as the pull-in instability.

6.1.1 Geometry of the problem

The geometry of the micro-beam, as well as the different boundary conditions of the coupled problem, are described in Figure 46. They are implemented in the `clampedMicroBeam.geo` file available on the master branch. Note that no mechanical surface traction is directly applied to the beam and in this case, gravity is neglected. The beam is actuated electrostatically, which means that a voltage difference ΔV is applied between the lower electrode and the silicon structure. The resulting electric field induces an electrostatic pressure on the surface of the beam, which acts as the only mechanical load of the problem. In this case, the electric field is expected to be far bigger (in norm) on the lower surface of the beam than on the top surface. The electrostatic pressure acting as a *pull* force, the beam will tend to be pulled down by the applied voltage difference.

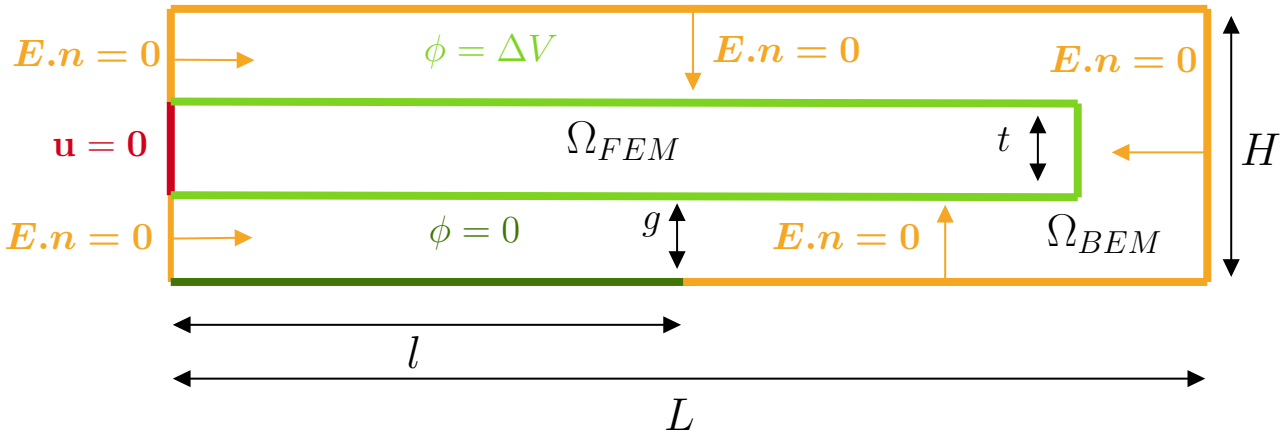
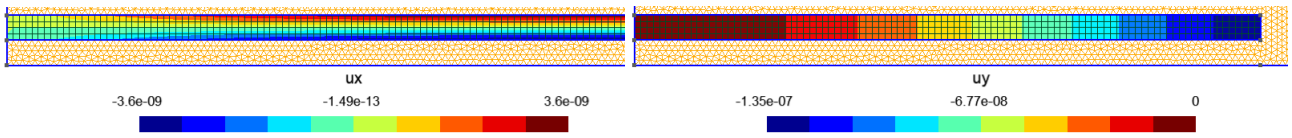


Figure 46: Geometry (not at scale) associated to one clamped micro-beam. The silicon beam, represented by the inside of the FEM domain Ω_{FEM}, is clamped on its left edge corresponding to an imposed zero displacement $u_x = u_y = 0$. Its Young's modulus is assumed to be $E = 150$ [GPa] and its Poisson ratio $\nu = 0.27$ [-]. Its length is fixed to $l = 25$ [µm] and its thickness to $t = 1$ [µm]. The external boundary of the beam (in light green) is supposed to be conductive and is set to $\phi = \Delta V$ (adjustable), while the lower electrode (in dark green) is the mass ($\phi = 0$ [V]). The gap between the beam and the lower electrode is chosen to be $g = 1$ [µm]. On the remaining boundary of the BEM domain (in orange), the normal component of the electric field is set to $\boldsymbol{E} \cdot \boldsymbol{n} = 0$ [V/m], so that it does not interfere with the electrostatics of the micro-system. Air is considered in the Ω_{BEM} domain, so that $\varepsilon = \varepsilon_0 = 8.85 \cdot 10^{-12}$ [F/m].

6.1.2 Comparison between the linear and the non-linear solver

In this section, the results obtained with both implemented coupled solvers are described. In particular, the focus is set on the displacement field.
For the sake of comparison, a voltage difference of $\Delta V = 90$ [V] is applied. First order elements are used in the whole domain. The Ω_{FEM} domain is discretized using 400 quadrangular, while the boundary of Ω_{BEM} domain is composed of 520 linear elements. The inside of the Ω_{BEM} domain is filled with triangular elements. The displacement field obtained with the linear solver is represented in Figure 47. As can be observed, the displacement field in the beam corresponds roughly to a flexion configuration. The upper part of the beam is stretched and the lower part is compressed. The maximal vertical displacement is observed at the tip of the structure.

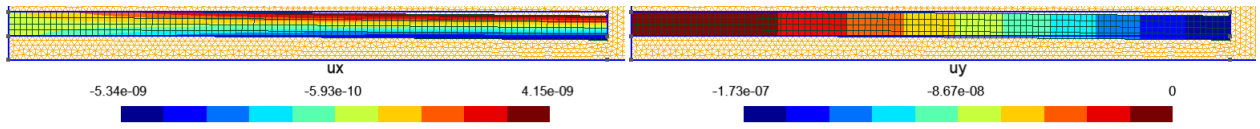


(a) Horizontal displacement field u_x [m]. (b) Vertical displacement field u_y [m].

Figure 47: Displacement field u [m] obtained with the linear solver for the micro-beam actuated electrostatically described in Figure. 46 with $\Delta V = 90$ [V].

However, as the vertical displacement at the tip of the beam $u_y = 0.135$ [µm] corresponds to 13.5 [%] of the gap between the structure and the electrode, the electric potential field is perturbed by the displacement of the beam and the iterative non-linear solver is required in order to obtain physically accurate results. The displacement field obtained with the iterative solver is represented in Figure 48. The numerical results have been obtained after 13 [-] iterations of the non-linear coupled solver, for a total CPU time of 1.959 [s]. For the sake of comparison, the linear solver takes 0.443 [s].

The shape of the displacement field is similar, but the maximal vertical displacement is now $u_y = 0.173$ [µm], which is quite different from the previous result. It implies that the non-linear iterative solver is best suited to study the considered problem.



(a) Horizontal displacement field u_x [m]. (b) Vertical displacement field u_y [m].

Figure 48: Displacement field u [m] obtained with the non-linear iterative solver for the micro-beam actuated electrostatically described in Figure. 46 with $\Delta V = 90$ [V]. It is displayed on the deformed final configuration.

6.1.3 Resulting stress and electric potential fields

Before focussing on the stress and electric potential fields, some simulations are performed when refining the mesh, in order to make sure the numerical results have converged towards the physical ones. The results are gathered in Table 6. As can be seen, using 2500 elements in the Ω_{FEM} domain is enough to ensure convergence.

Number of elements [-]	Maximal vertical displacement u_y [µm]
25	0.110
100	0.155
400	0.173
900	0.177
1600	0.179
2500	0.180
3600	0.180

Table 6: Convergence study for the clamped micro-beam geometry with $\Delta V = 90$ [V], using the non-linear iterative solver. The number of elements is the total number of elements in the Ω_{FEM} domain.

The resulting von Mises stress field is represented in Figure 49. The stress is the highest at the clamped edge of the beam, while the tip of the structure is almost stress free. It corresponds once again to a configuration similar to bending. Note that the maximal von Mises stress is far lower than the yield stress of silicon, which is $\sigma_{\text{y}} = 165$ [MPa], so that the behaviour is indeed elastic.

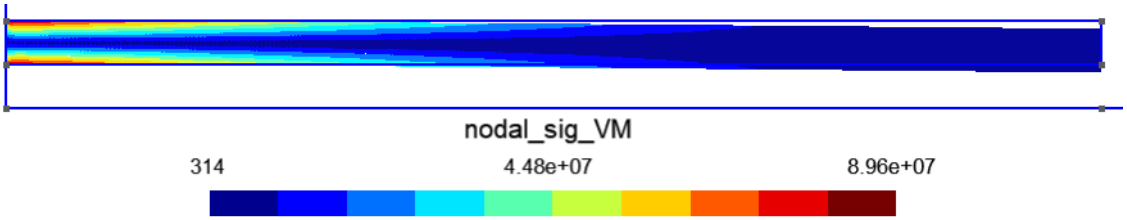


Figure 49: Equivalent von Mises stress field σ^{VM} [Pa] obtained with the non-linear iterative solver for the micro-beam actuated electrostatically described in Figure. 46 with $\Delta V = 90$ [V]. It is displayed on the deformed final configuration and computed with 2500 [-] elements inside the Ω_{FEM} domain.

The resulting electric potential field is represented in Figure 50. One can observe as the field adapts to the bending of the microbeam, the equipotential lines are not horizontal as it would be the case for a plane capacitor. Once again, this is due to the two-way coupling between the electrostatics and the displacement of the beam.

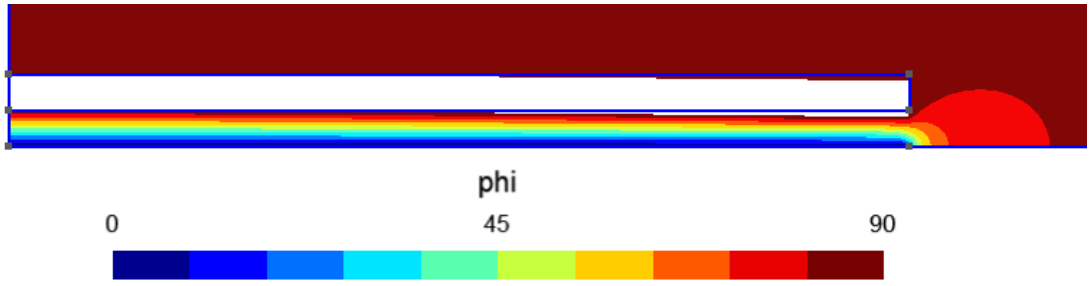


Figure 50: Electric potential field ϕ [V] obtained with the non-linear iterative solver for the micro-beam actuated electrostatically described in Figure. 46 with $\Delta V = 90$ [V]. It is displayed and computed on the deformed final configuration.

6.1.4 Voltage-displacement curve

The maximal vertical deflection of the micro beam can be computed as a function of the applied voltage difference. The results are represented in Figure 51. As can be observed, the linear one-way coupled solver and the non-linear two-way coupled solver provide very similar results below $\Delta V = 50$ [V]. For greater voltage differences, the displacement of the beam impacts the distribution of the electric potential below the beam. In this case, the non-linear solver is required to obtain realistic results.

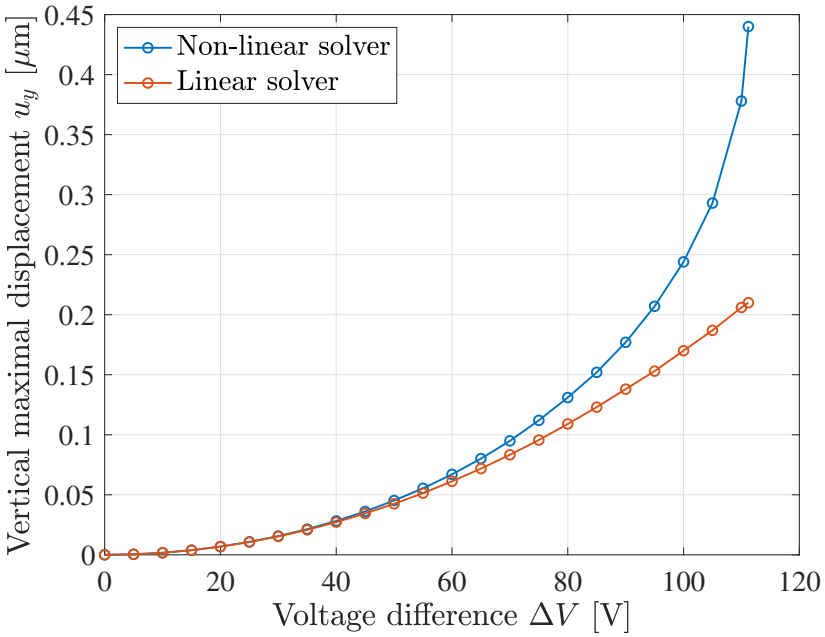


Figure 51: Evolution of the maximal vertical displacement u_y (in absolute value and in [µm]) as a function of the voltage difference ΔV [V] for the micro beam described in Figure 46. Results obtained with both the one-way coupled linear solver and the two-way coupled non-linear solver, using 900 square elements in the FEM domain Ω_{FEM}.

As can be observed for the the one-way linear solver, the maximal vertical deflection of the micro beam is a quadratic function of the voltage. This behaviour is a characteristic of the electrostatic actuators in small displacement configurations using one single electrode as a mechanical charge. The small displacement assumption implies that the displacement of the structure does not impact the electrostatics distribution.

Indeed, if the electric potential imposed on the electrode ΔV is the only non-homogeneous boundary condition on the boundary of the BEM domain Ω_{BEM}, the electric potential field ϕ and the electric field $\boldsymbol{E}$ are proportional to ΔV. This is a direct result from the superposition theorem, which holds as Laplace equation (Equation 19) is linear. Hence, the electric field on the boundary between the FEM and the BEM domains is directly proportional to ΔV. The electrostatic pressure, related quadratically to the electric field through Equation 70, is hence proportional to $(\Delta V)^2$. If the electrostatic pressure is the only mechanical charge applied on the structure, the superposition theorem (linear elasticity is assumed) states that the displacement field is proportional to the electrostatic pressure and therefore proportional to $(\Delta V)^2$.
Note that this result is only valid if the displacement of the structure does not modify the

electrostatics distribution.

As can be observed in Figure 51, the maximal displacement increases rapidly for an applied voltage $\Delta V > 50$ [V] and the non-linear solver is required. The number of iterations required for the iterative solver to converge (until a certain tolerance) also increases as represented in Table 7 and Figure 52. This can be explained by the fact the bigger is the voltage difference, the greater is the displacement and the greater is the impact of the displacement on the geometry of the electrostatic problem. As a result, the two physics of the coupled problem strongly interact with each other and a higher number of iterations is required for the solver to reach the equilibrium configuration. Figure 52 also shows that the relative displacement difference still exponentially decays with the number of iterations.

Voltage difference ΔV [V]	Number of iterations [-]
50	7
60	8
70	9
80	11
90	14
100	18
110	50

Table 7: Number of iterations required for the non-linear iterative solver to reach a relative displacement difference (Equation 95) of 10^{-7} for given voltage differences ΔV [V], in the clamped beam configuration of Figure 46.

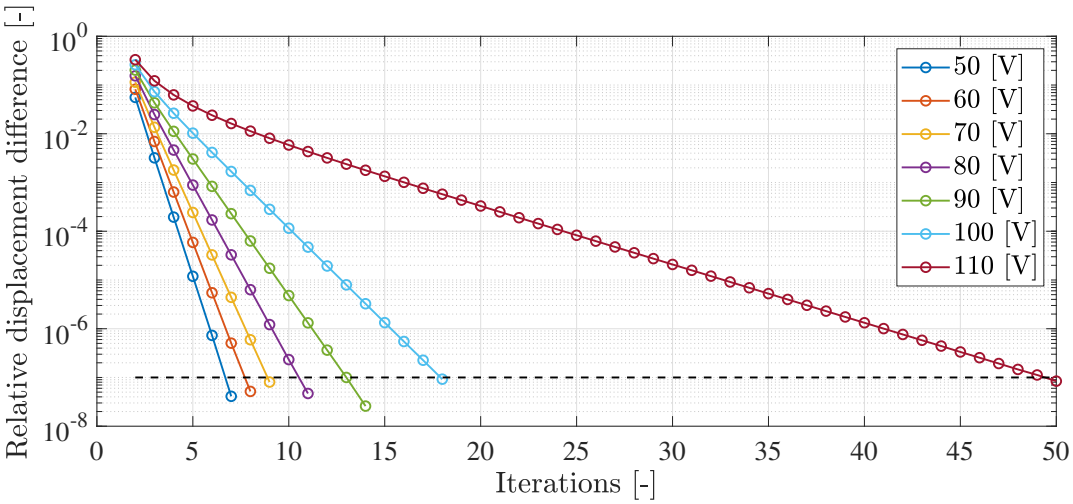


Figure 52: Evolution of the relative displacement difference as a function of the number of iterations for different voltage differences ΔV [v], in the clamped beam configuration of Figure 46. The threshold relative displacement difference has been fixed to 10^{-7}.

6.1.5 Pull-in voltage

The pull-in voltage ΔV_{pi} [V] is defined as the minimal voltage difference for which the micro-beam is deflected in such a way that it sticks to the bottom electrode. By definition, such a

configuration requires large displacements, which is why only the non-linear iterative solver is considered in this section.

As contact between different surfaces has not been implemented, such a deflection is characterized by a simple divergence of the iterative algorithm when the iterative solver is executed.

For the geometry considered in Figure 46, the highest voltage difference that does not induce a divergence of the iterative solver, hence corresponding to the pull-in voltage, is $\Delta V_{\text{pi}} = 111.1$ [V]. It has been computed using 900 elements in the Ω_{FEM} domain.
An example of a diverged solution is represented in Figure 53.

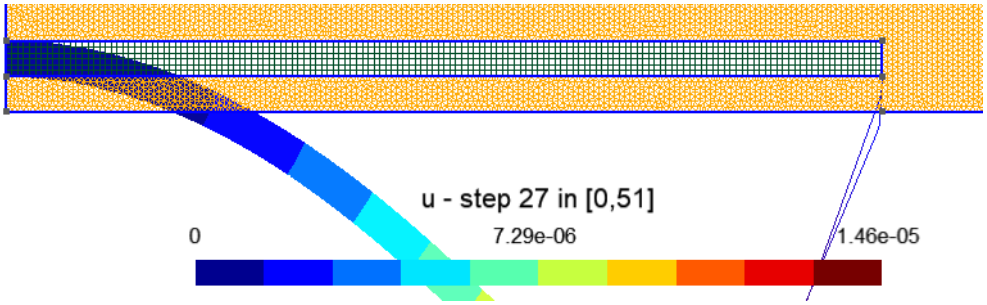


Figure 53: Deformed configuration obtained after iteration 27 for the clamped micro-beam, for an applied voltage difference $\Delta V = 112$ [V]. The result does not represent the physics, as the iterative solver has diverged.

In [6], they developed an approximate analytical formula for the pull-in voltage for such a clamped micro-beam, by assuming a linear shape between the tip and the anchor of the structure:

$$\Delta V_{\text{pi}} = \sqrt{0.22 \frac{E g^3 t^3}{\varepsilon l^4}}, \tag{96}$$

corresponding in the present case to $\Delta V_{\text{pi}} = 97.702$ [V]. However, as can be observed in Figure 49, the vertical deflection of the beam can not be approximated by a straight line. Moreover, according to [6], the linear deflection assumption underestimates the pull-in voltage.
In [7], they studied the problem numerically by taking the non-uniform electrostatic pressure acting on the deflected beam into account and they obtained an other formula for the pull-in voltage in the current configuration:

$$\Delta V_{\text{pi}} = \sqrt{0.28 \frac{E g^3 t^3}{\varepsilon l^4}}, \tag{97}$$

corresponding now to $\Delta V_{\text{pi}} = 110.224$ [V]. This result is very close to the one obtained in this report, corresponding to a relative error of 0.8 [%]. This result further validates the implemented iterative non-linear FEM-BEM solver.

6.2 Longitudinal comb-drive accelerometer

Comb-drive devices are used for the electrostatic actuation of microsystems in many different applications from micromirrors to accelerometers. They rely on interdigitated fins with an applied voltage difference inducing strong electric potential gradients in the free space between the fins. The strong gradients result in a large electric field and a large electrostatic pressure acting on the moving electrode. In the present case, the device is first studied as an actuator and later as an accelerometer.

6.2.1 Geometry of the device

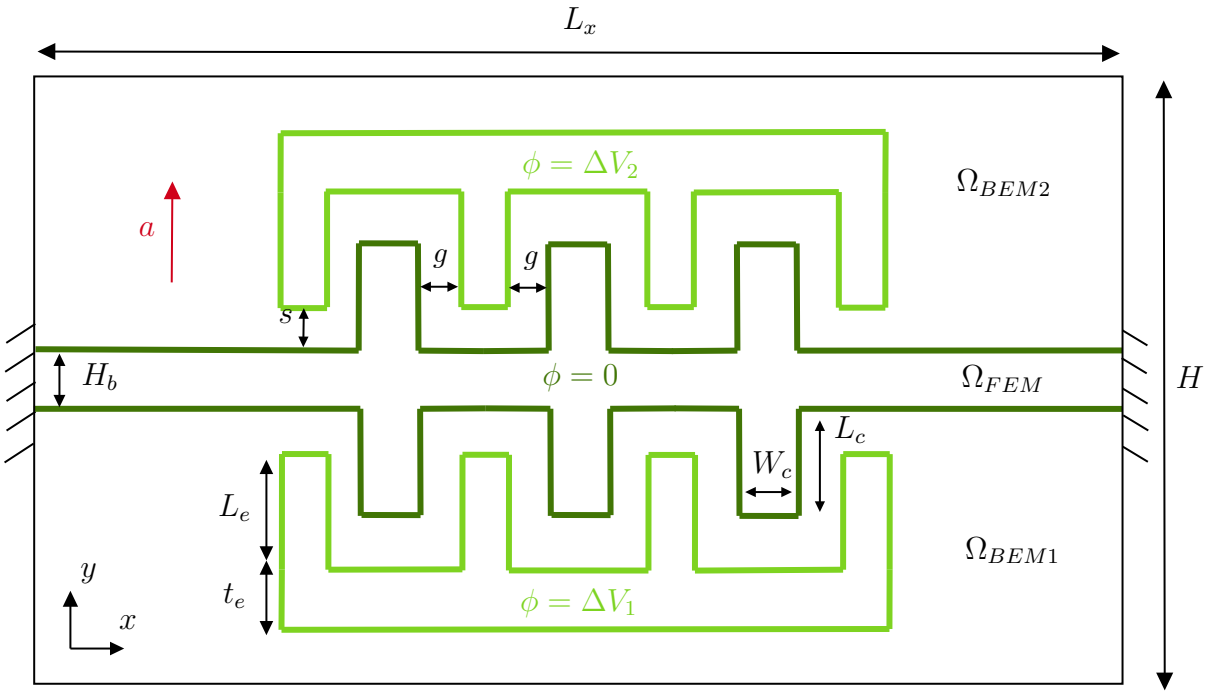


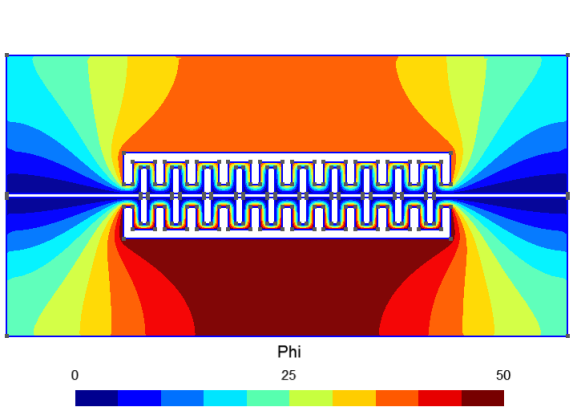
Figure 54: Geometrical description of a comb-drive actuator / accelerometer, implemented in the `longitudinalCombDevice.geo` file. The bottom and the top electrode are fixed and are respectively subjected to a potential difference ΔV_1 and ΔV_2 with respect to the central beam acting as the mass of the system ($\phi = 0$ [V]). The central beam of length $L_x = 120$ [µm] is clamped at both ends. The height of the numerical domain is $H = 60$ [µm]. An homogeneous Neumann condition (normal component of the electric field) is applied on the remaining boundary of the domain. The present device has $N_c = 3$ [-] fingers on each side of the beam. The beam is made of silicon: $E = 150 \cdot 10^9$ [Pa], $\nu = 0.27$ [-], $\rho = 2300$ [kg/m^3]. The height of the central beam is $H_b = 0.8$ [µm], the length of one fin is $L_c = 5.6$ [µm] and its width $W_c = 1.6$ [µm]. The horizontal space between the fins of the electrodes and the central fins is $g = 1.6$ [µm]. The electrode fin length is $L_e = 4.8$ [µm], while their width is equal to the thickness of the electrode basis $t_e = 2$ [µm]. The vertical space between the central beam and the tip of the fins of the electrodes is $s = 2$ [µm]. Air is located between the different electrodes: $\varepsilon = \varepsilon_0 = 8.85 \cdot 10^{-12}$ [F/m]. a [m/s^2] is the vertical acceleration experienced by the central beam.

The geometry of the comb-drive device is presented in Figure 54. Note that the number N_c of fins as well as all dimensions can be chosen by the user in the `longitudinalCombDevice.geo` file. The bottom and the top electrode are fixed and the central beam plays the role of the moveable electrode. When used as an accelerometer, the central beam plays the role of

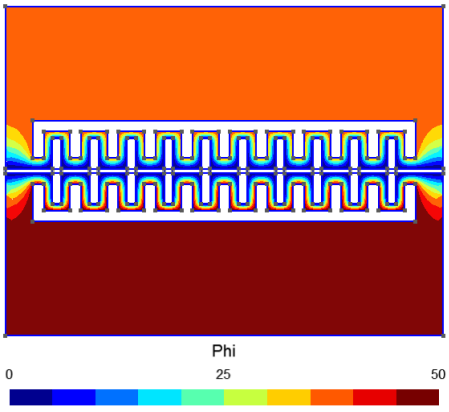
the proof mass. An homogeneous Neumann boundary condition is applied on the external boundary of both BEM domains. It can be justified as the electric field far away from the electrode should be equal to zero, as the electric potential variations should disappear far from the electrodes.

The first step of the numerical study of the comb-drive device is to check that the dimensions of the external domain do not modify the electrostatic configuration of the electrodes. As can be observed in Figure 55(b), the length of the numerical domain must be much larger than the length of the fixed electrodes, otherwise the lateral boundaries of the numerical domain impact the distribution of the potential field. The height of the numerical domain should not be chosen too small either, even if its impact on the potential field between the electrodes is less significant than in the case of the domain length, as can be seen in Figure 55(c). For the rest of the study, the reasonable configuration represented in Figure 55(a) is kept. Note that the electric field acting near the ends of the central beam is negligible with respect to the electric field at the top of the fins (one thousand times smaller). Note that the variations of the maximal vertical displacement obtained for the three different configurations are negligible (less than 0.1 [%] of relative variation), as the electric field between the electrodes is almost not impacted by the domain dimensions.

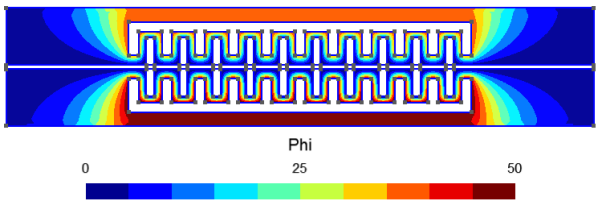
Please note the ability of the presented implementation to handle multiple independent BEM domains.



(a) $L_x = 120$ [µm], $H = 60$ [µm].



(b) $L_x = 80$ [µm], $H = 60$ [µm].



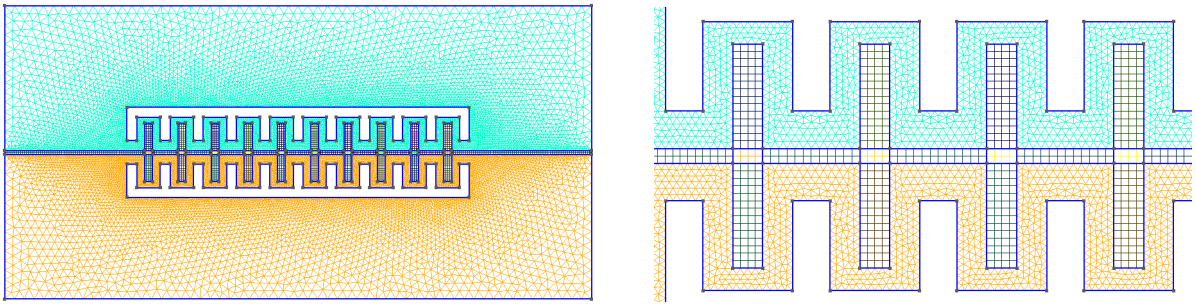
(c) $L_x = 120$ [µm], $H = 24$ [µm].

Figure 55: Electric potential ϕ [V] field obtained for the comb-device configuration described by Figure 54 with $N_c = 10$ fins, $\Delta V_1 = 50$ [V] and $\Delta V_2 = 40$ [V], for various numerical domain dimensions L_x and H. Results obtained with the iterative solver.

6.2.2 Mesh

The structure of the mesh used to perform the numerical simulations is represented in Figure 56. In the FEM domain, the mesh is structured and is composed of quadrangular elements, as triangular elements behave poorly in bending configurations as discussed in Sec-

tion 1.4. Moreover, the elements are chosen to be square to allow exact numerical integration using Gauss integration. Inside the BEM domain, the type of the elements does not matter for the accuracy as the elements are only used for post-processing. Hence, triangular elements are chosen as they adapt more easily to the complex geometry between the fins of the electrodes. In order to reduce the computation time, the size of the linear elements is increased on the external boundary of the BEM domain as the solution does not vary significantly and as the important part of the BEM computation is located at the surface of the electrodes. For this application, first order elements are chosen.



(a) Global view. (b) Zoom on the left part.

Figure 56: Mesh used for numerical simulations involving the comb-drive device, represented with $N_c = 10$ fins. Note that the mesh in the central beam is refined for performing numerical simulations.

6.2.3 Convergence of the maximal vertical displacement

In order to determine the required number of finite elements in the FEM domain, the convergence of the maximal vertical displacement as the mesh is refined is studied. The study is performed in the following configuration: $N_c = 4$ fins, $\Delta V_1 = 60$ [V], $\Delta V_2 = 0$ [V], $a = 0$ [m/s^2], using the non-linear iterative solver. Numerical results are gathered in Table 8. As can be seen, more than 100000 nodes are required in order to reach a relative error less than 1 [%]. However, it requires a lot of computation time. Such an important CPU time is needed because it takes about 20 iterations to reach the nodal difference tolerance of 10^{-7} and it takes approximately 8 steps for the single FEM solver to converge during one iteration. Hence, using $n_{\text{FEM}} = 40174$ nodes is satisfactory enough (the relative error is lower than 2 [%]). It corresponds to 2016 square finite elements in each fin.

N_{BEM} [-]	n_{FEM} [-]	Maximal vertical displacement u_y [µm]	CPU time [s]
1548	1451	0.250	3.75
2374	5006	0.290	10.28
4022	18398	0.308	33.41
5670	40174	0.314	94.55
7318	70334	0.318	162.51
8966	108878	0.319	246.29

Table 8: Convergence study of the absolute value of the maximal vertical displacement for the comb-device with $N_c = 4$ fins, $\Delta V_1 = 60$ [V], $\Delta V_2 = 0$ [V], $a = 0$ [m/s^2], using the non-linear iterative solver. The number of nodes n_{FEM} is the total number of nodes in the Ω_{FEM} domain, while the number of elements N_{BEM} is the total number of linear elements on the Ω_{BEM} domain boundary.

Note that the gravity can be neglected as the maximal vertical displacement is $3.47 \cdot 10^{-6}$ [µm] for $N_c = 4$ fins, $\Delta V_1 = 0$ [V], $\Delta V_2 = 0$ [V], $a = g = 9.81$ [m/s^2].

The convergence of the iterative solver as a function of the number of iterations can also be studied for this application. The results are shown in Figure 57 for different potentials ΔV_1 below pull-in. As for the other applications, one can clearly see the exponential convergence of the relative displacement difference.

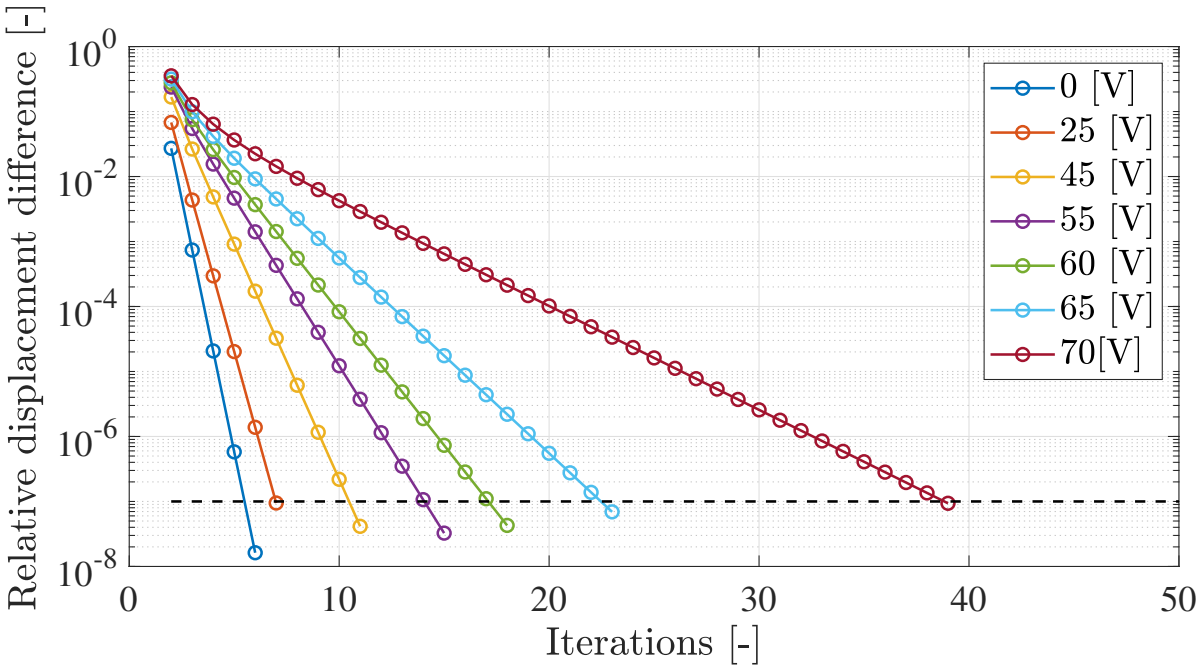
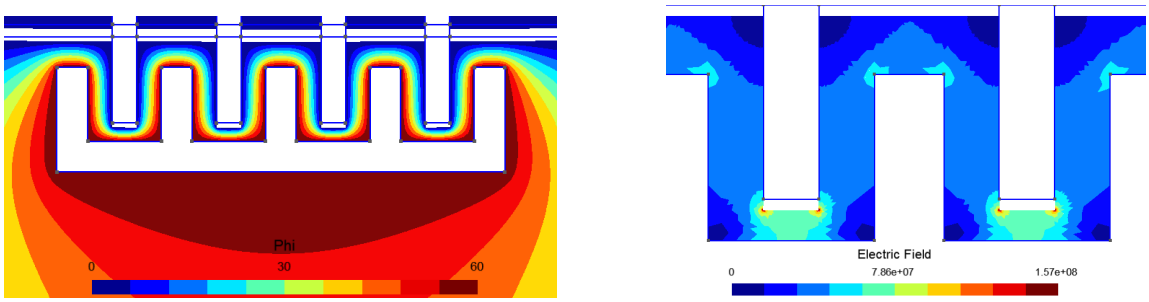


Figure 57: Evolution of the relative displacement difference as a function of the number of iterations of the iterative solver for the longitudinal comb-drive actuator in the configuration described in Figure 54, with $N_c = 4$ fins, $\Delta V_2 = 20$ [V], $a = 0$ [m/s^2] and ΔV_1 variable.

6.2.4 Different physical fields

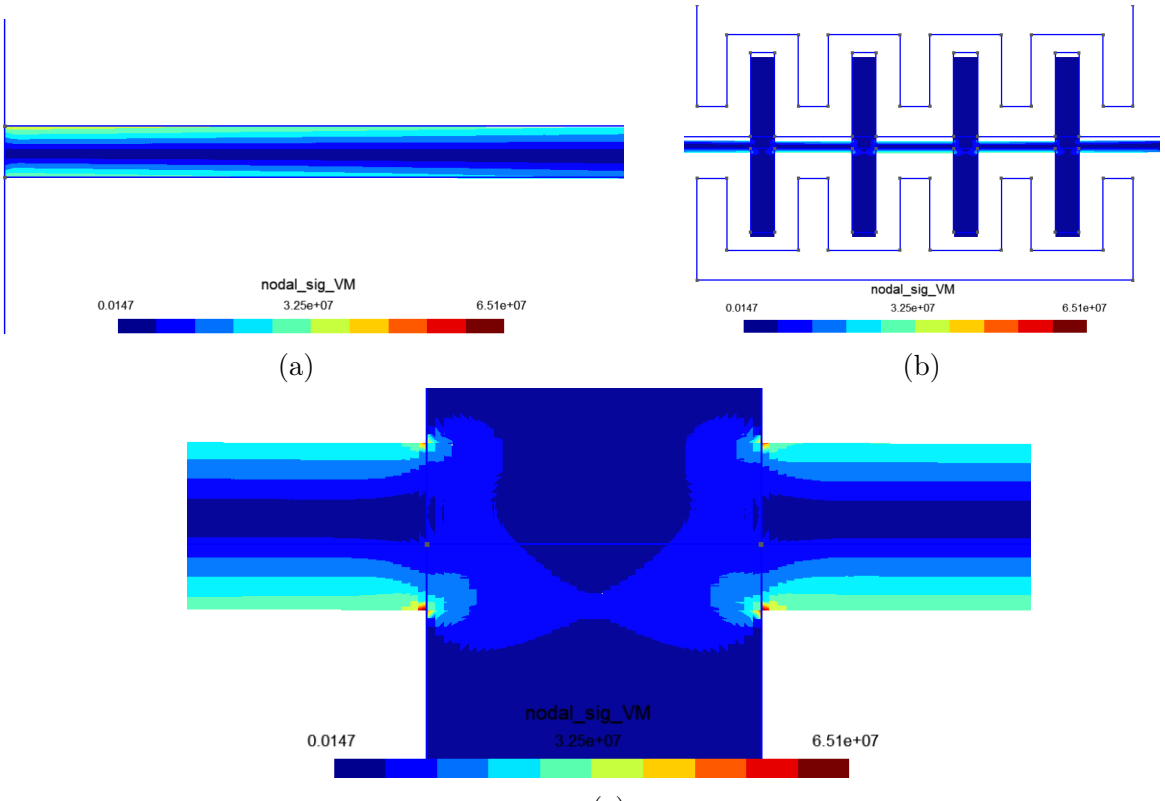
It is interesting to observe the distribution of different fields computed numerically. In this section, the results are presented in the configuration from last section ($N_c = 4$ fins, $\Delta V_1 = 60$ [V], $\Delta V_2 = 0$ [V], $a = 0$ [m/s^2]), using $n_{\text{FEM}} = 108878$ elements in the FEM domain.
The electric potential and the electric field between the electrodes are represented in Figure 58. As can be observed, the equipotential lines follow the geometry of the electrodes and are squeezed by the vertical displacement of the central beam. The potential varies from 0 [V] to 60 [V] over distances smaller than 2 [µm], resulting in a large electric field of the order of magnitude of $\|\boldsymbol{E}\| \approx 60/2 = 30$ [V/µm] $= 3 \cdot 10^7$ [V/m]. As the vertical space between the base of the central beam and the fins of the bottom electrode is 0.8 [µm] larger than the space between the central fins and the base of the bottom electrode in the undeformed configuration, it is more than 1 [µm] larger in the present deformed configuration and the electric field is smaller at this location. However, in the horizontal space between the moving fins and the fixed fins, the electric field is almost uniform and horizontal (resulting from almost vertical equipotential lines). Locally, the geometry corresponds to the configuration of a plane capacitor.



(a) (b)

Figure 58: (a) Electric potential ϕ [V] and (b) amplitude of the electric field [V/m] computed for the comb-drive device using the non-linear iterative solver, with $N_c = 4$ fins, $\Delta V_1 = 60$ [V], $\Delta V_2 = 0$ [V], $a = 0$ [m/s^2] and $n_{\text{FEM}} = 108878$ elements in the FEM domain.

The von Mises stress field in the central beam is represented in Figure 59. At the clamped ends of the beam, the stress field is similar to a bending configuration as the neutral axis of the beam is stress free. The fins are almost stress free, however one suspects a singularity of the stress field at the corner between the central beam and the fins. Such a divergence at corners is also encountered in macroscopic mechanics and it could be removed by introducing fillets, as explained in [1]. However, note that the maximal stress is far lower than the initial yield stress of silicon $\sigma_{\text{y}}^0 = 165$ [MPa], so that the elastic assumption is still valid.



(a) (b)

(c)

Figure 59: Von Mises stress field σ_{VM} [Pa] computed for the comb-drive device using the non-linear iterative solver, with $N_c = 4$ fins, $\Delta V_1 = 60$ [V], $\Delta V_2 = 0$ [V], $a = 0$ [m/s^2] and $n_{\text{FEM}} = 108878$ elements in the FEM domain. It is represented (a) at the left clamped edge, (b) in the central part of the beam and (c) zoomed on the base of one fin.

6.2.5 Voltage-displacement curve for the actuator

From an engineering point of view, the most interesting relation to characterize is the relation between the voltage difference ΔV between an electrode and the central beam and the induced vertical displacement. First, the impact of ΔV_1 with $\Delta V_2 = 0$ [V] is studied and the resulting curve is represented in Figure 60. The one-way coupled linear solver and the two-way coupled iterative solver are compared, and once again the results are very similar for small voltage differences. As discussed in Section 6.1.4, the curve obtained for the linear solver is quadratic as for the linear solver the electric field is only computed on the undeformed configuration. However, for large voltage differences, the results obtained with the non-linear solver, taking the physical coupling between the electrostatics and the displacement of the beam into account, are more accurate. As can be observed, the vertical displacement increases abruptly as the voltage difference reaches $\Delta V_1 = 72$ [V], corresponding to the pull-in voltage of the comb device. When the voltage difference is greater than the pull-in voltage, the central beam sticks to the bottom fixed electrode. Note that the exact same curve can be obtained by setting $\Delta V_1 = 0$ and by varying ΔV_2. In this case, the vertical displacement is in the other direction.

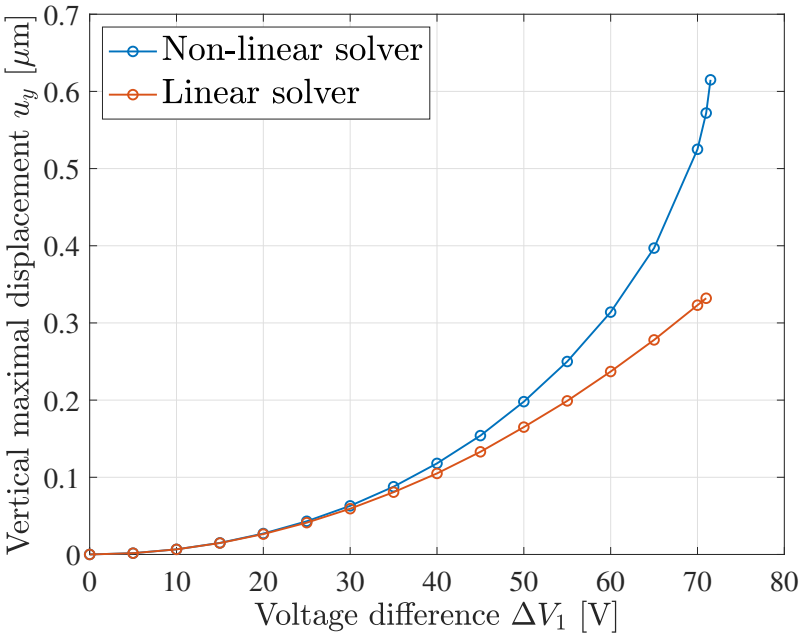


Figure 60: Evolution of the maximal vertical displacement u_y (in absolute value and in [µm]) as a function of the voltage difference ΔV_1 [V] applied on the bottom electrode of the comb device described in Figure 54, with $N_c = 4$ fins, $\Delta V_2 = 0$ [V] and $a = 0$ [m/s^2]. Results obtained with both the one-way coupled linear solver and the two-way coupled non-linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

The relation between the applied voltage difference and the maximal vertical displacement can also be retrieved for a non-zero voltage difference ΔV_2 between the upper electrode and the central beam. In Figure 61, the results obtained with $\Delta V_2 = 20$ [V] are compared to the previous results. As can be observed, when applying a voltage difference on the upper electrode, the vertical displacement of the central beam is lower than in the previous case. By symmetry, the vertical displacement is equal to zero for $\Delta V_1 = \Delta V_2 = 20$ [V]. The shape of both curves are very similar and the pull-in voltage is nearly the same, independently from

ΔV_2, provided $\Delta V_2 \ll \Delta V_1$. This is expected as the force exerted on the central beam is in first approximation proportional to the square of the voltage difference.
When comparing the behaviour of the vertical displacement as a function of the total voltage difference between the upper and lower electrodes, the pull-in instability is reached sooner when ΔV_2 is larger

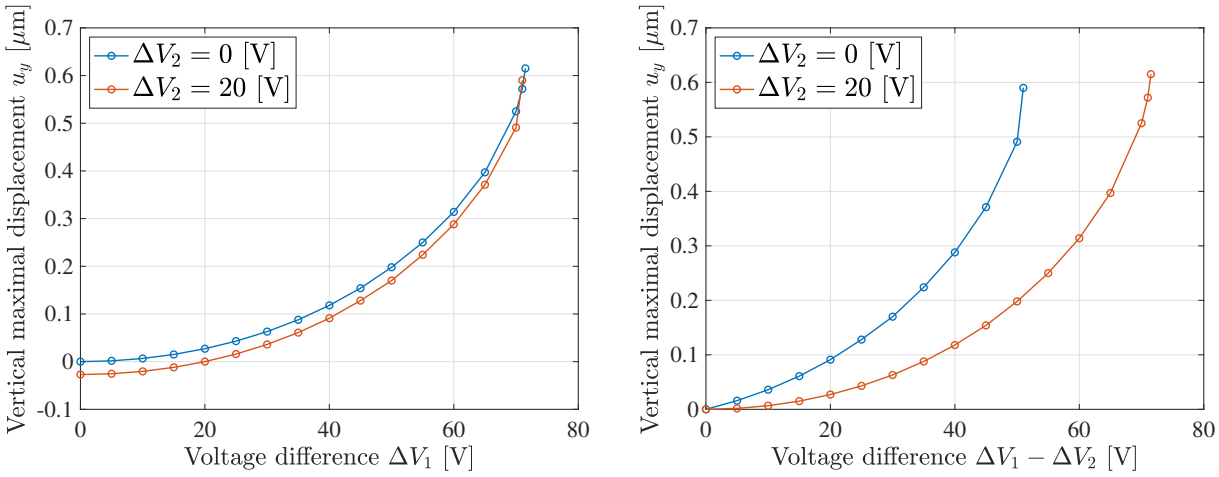


Figure 61: Evolution of the maximal vertical displacement u_y [µm] as a function of (a) the voltage difference ΔV_1 [V] applied on the bottom electrode of the comb device described in Figure 54; (b) the voltage difference $\Delta V_1 - \Delta V_2$ [V] applied between the bottom and top electrodes, with $N_c = 4$ fins and $a = 0$ [m/s^2], for $\Delta V_2 = 20$ [V] and $\Delta V_2 = 0$ [V]. Results obtained with the two-way coupled non-linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

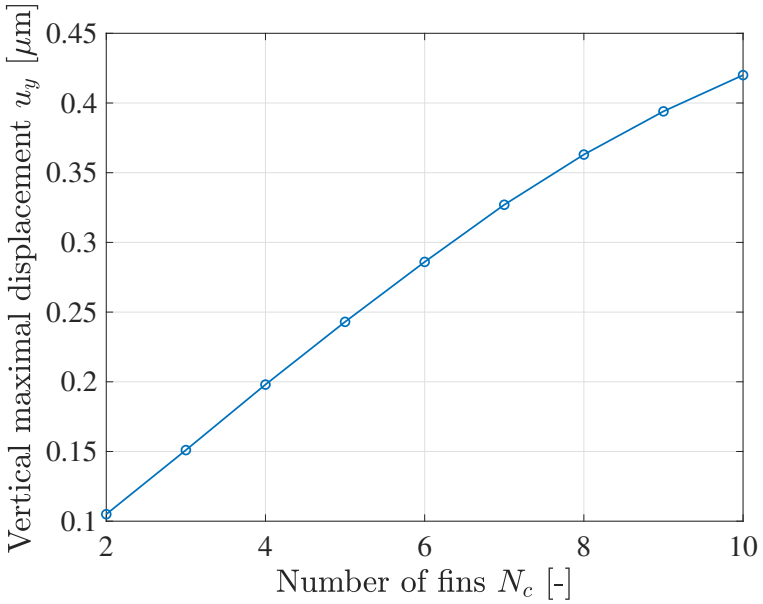


Figure 62: Evolution of the maximal vertical displacement u_y (in absolute value and in [µm]) as a function of the number of fins N_c of the comb device described in Figure 54, with $\Delta V_1 = 50$ [V], $\Delta V_2 = 0$ [V] and $a = 0$ [m/s^2]. Results obtained with the two-way coupled non-linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

Moreover, the impact of the number of fins N_c under constant voltage differences $\Delta V_1 = 50$ [V] and $\Delta V_2 = 0$ [V] is studied and represented in Figure 62. As expected, the vertical displacement increases with the number of fins, as the strongest electric field is applied on the fins and the resulting electrostatic actuation is the most effective on the fins.

For a small number of fins, resulting in small displacements, the dependence seems to be linear. This can be explained using an approximate theoretical development introduced in [8]. In the first approximation, the total force exerted by the electric field on one fin can be split into two components.

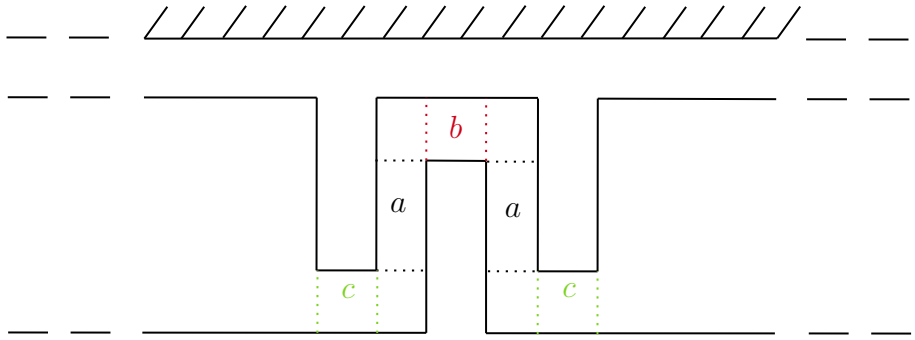


Figure 63: Representation of the different force components around one fin of the moving central beam (at the bottom) and the fixed electrode (at the top). The lateral zone (a) can be approximated as a plane capacitor, as well as the front zone (b) and the base zone (c).

First, a lateral force resulting from the fact that the system tries to increase the overlap of the moving and fixed fins (zone (a) in Figure 63) as it is favorable from an energetic point of view, can be derived assuming the configuration of a plane capacitor:

$$F_a = 2 \cdot \frac{\varepsilon \Delta V^2}{2g}, \tag{98}$$

in which the factor 2 corresponds to the two lateral faces of the fin. The second component of the force is due to the approximate plane capacitor configuration between the top of the moving fin and the base of the fixed electrode (zone (b) in Figure 63):

$$F_b = \frac{\varepsilon W_c \Delta V^2}{2s_e^2}, \tag{99}$$

with $s_e = s + L_e - L_c = 1.2$ [µm] the distance between the top of the moving fin and the base of the fixed electrode. In the first approximation, the total force exerted by the fixed electrode on the central beam (with N_c fins) is then:

$$F = N_c \cdot \varepsilon \left(\frac{1}{g} + \frac{W_c}{2s_e^2} \right) \Delta V^2 + (N_c + 1) \varepsilon \frac{W_c}{2s^2} \Delta V^2, \tag{100}$$

in which the last term corresponds to the force F_c resulting from the field between the top of the electrode fins and the base of the central beam (zone (c) in Figure 63). Finally, by considering that the central beam has a constant rigidity (it is discussed a bit later), the vertical displacement is proportional to the vertical electrostatic force and hence proportional to the number of fins. Note that this approximation is only valid for a small number of fins / for small displacements, as for a greater number of fins the force is not only applied on the

middle point of the beam (which breaks the constant rigidity approximation) and a greater number of fins induces large displacements which are more complex to describe.

One can also study the evolution of the pull-in voltage difference ΔV_1 as a function of the number N_c of fins, for $\Delta V_2 = 0$ [V]. As can be observed in Figure 64, the pull-in voltage is decreased when the number of fins is increased. This is once again expected: as explained above, increasing the number of fins induces a greater force on the central beam, even if ΔV_1 is kept constant.

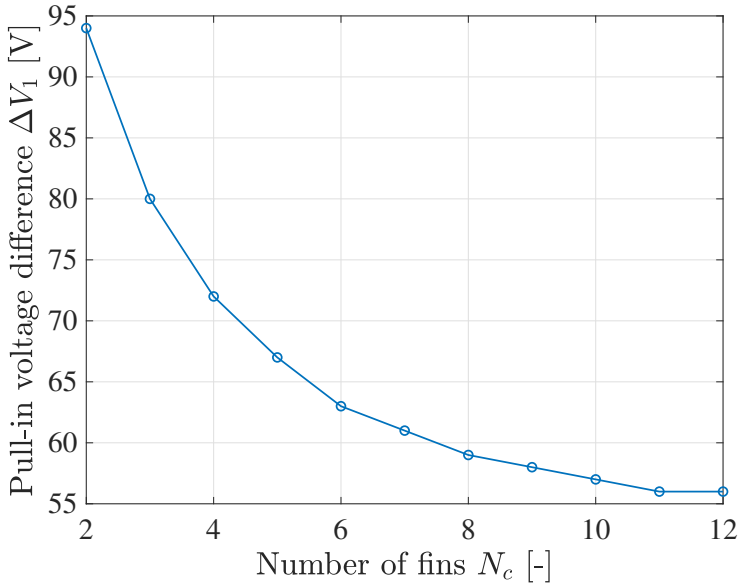


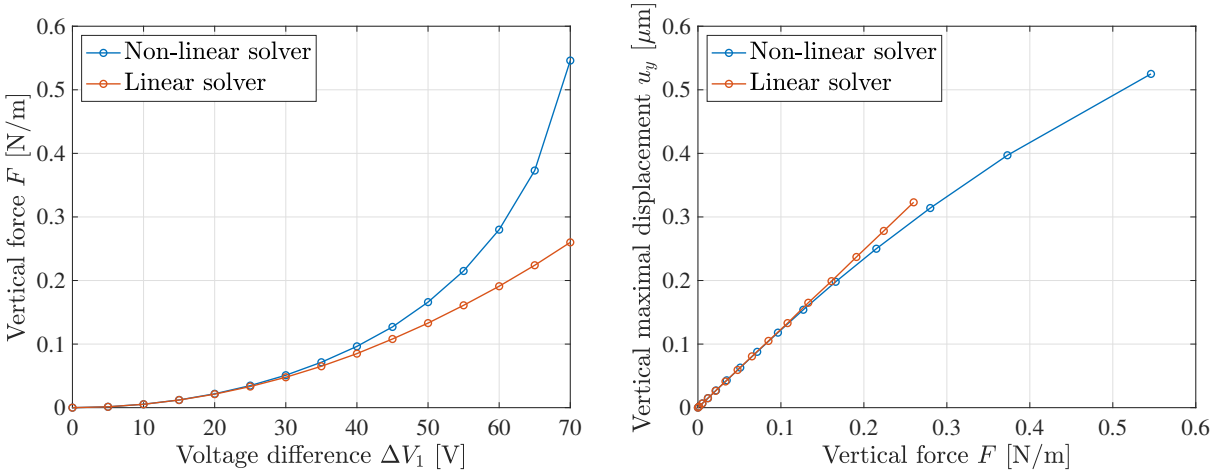
Figure 64: Evolution of the pull-in voltage difference ΔV_1 [V] applied on the bottom electrode as a function of the number of fins N_c of the comb device described in Figure 54, $\Delta V_2 = 0$ [V] and $a = 0$ [m/s^2]. Results obtained with the two-way coupled non-linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

The coupling between the electrostatics and the kinematics of the structure is entirely due to the force resulting from the electrostatic pressure. The relation between the voltage difference and the electrostatic force applied to the central beam is represented in Figure 65(a). As explained in Section 6.1.4, the force depends quadratically on the voltage difference in the small displacements configuration as it is always the case for the linear one-way coupled solver. The same conclusion can be drawn from Equation 100. For larger displacements, the deflection of the moving electrode impacts the distribution of the electric potential field and the force is larger than the linear prediction.

In a second phase, the electrostatic force induces a vertical displacement of the central beam. In the small displacements configuration (always assumed by the linear solver), the vertical deflection is proportional to the applied force, due to the linearity of the equations and the superposition theorem. This result can be observed in Figure 65(b). In fact, the proportionality constant between the force and the displacement is called the rigidity k [N/m^2], defined such that $F = k \cdot u_y$. The rigidity obtained numerically with the linear solver is equal to $k = 8.05 \cdot 10^5$ [N/m^2], which is of the same order of magnitude than the rigidity of a clamped-clamped beam with a point force exerted in its middle:

$$k_{\text{th}} = \frac{16EH_b^3}{L_x^3} = 7.11 \cdot 10^5 \text{ [N/m}^2\text{]}, \tag{101}$$

predicted by mechanics of materials [9].



(a) (b)

Figure 65: Evolution of (a) the vertical force F [N/m] applied on the central beam as a function of the voltage difference ΔV_1 [V] applied on the bottom electrode of the comb device described in Figure 54, with $N_c = 4$ fins, $\Delta V_2 = 0$ [V] and $a = 0$ [m/s^2]; (b) the maximal vertical displacement (in absolute value and in [µm]) as a function of the vertical force F [N/m] applied on the central beam. Results obtained with both the one-way coupled linear solver and the two-way coupled non-linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

6.2.6 Calibration of the accelerometer

The comb-drive device can also be used as an accelerometer, converting a vertical acceleration a into a voltage difference ΔV. This type of active sensor is described in [8] and relies on following principle: when subjected to a constant vertical acceleration, the central beam acts as a proof mass and is deflected. The voltage difference ΔV_1 and ΔV_2 are controlled and adjusted using a feed-back loop in order to induce an electrostatic pressure to cancel out the vertical deflection induced by the acceleration. In practice, if the central beam is deflected towards the top fixed electrode (see Figure 54), only the lower electrode is activated while the top voltage difference ΔV_2 remains equal to zero.

The calibration of the comb-drive device used as an accelerometer is performed numerically and the results are gathered in Figure 66 using logarithmic axes. As can be observed, the voltage difference ΔV_1 is proportional to the square root of the vertical acceleration a. Conversely, one can write $a \propto \Delta V^2$. Note that in practice, such an *ADXL* accelerometer is used for measuring accelerations below $100g$ to $1000g$ (500 to 10000 [m/s2]), depending on the dimensions, to avoid an unstable behaviour.

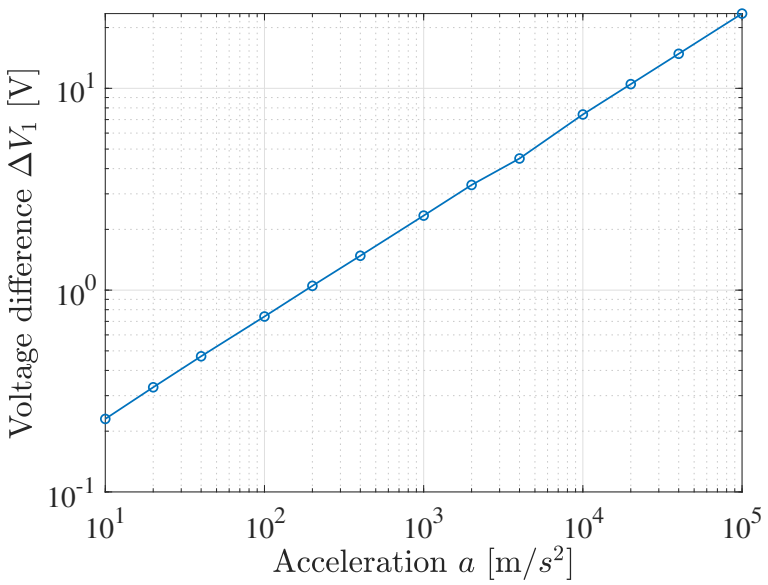


Figure 66: Relation between the vertical acceleration a [m/s^2] applied on the central beam and the required voltage difference ΔV_1 [V] applied on the bottom electrode in order to cancel the vertical displacement of the comb-device, for $\Delta V_2 = 0$ [V] and $N_c = 4$ [fins]. Results obtained with the one-way coupled linear solver, using 40174 nodes and square first order elements in the FEM domain Ω_{FEM}.

One main advantage of such a device is the predictability of the involved physical mechanisms. Indeed, as the deflection of the beam is controlled to be equal to zero, the displacements can not be large and the mechanical behaviour of the beam is linear. As explained in Section 6.1.4 (the small displacements assumption is valid in the present context), the electrostatic force exerted on the central beam is proportional to ΔV^2 and by symmetry, the force must be vertical. The second force acting on the system is the force induced by the constant vertical acceleration, which is also vertical and directly proportional to the acceleration a. Finally, by imposing the vertical deflection of the beam to be equal to zero, the net vertical force acting on the beam should also be equal to zero (as the vertical deflection is proportional to the net vertical force exerted to the beam, as explained in previous section). Hence, the electrostatic force proportional to ΔV^2 must balance the acceleration force proportional to a, which means the vertical acceleration a must be directly proportional to the square of the controlled voltage difference ΔV^2.

6.3 Folded flexure beam

The last application studied in this report is the folded flexure device, which is slightly adapted from [10]. It is mainly used as an electrostatic actuator in MEMS, but it can also be used as a accelerometer (only as a one-directional accelerometer as there is one single fixed controllable electrode). The related geometry is presented in Figure 67.

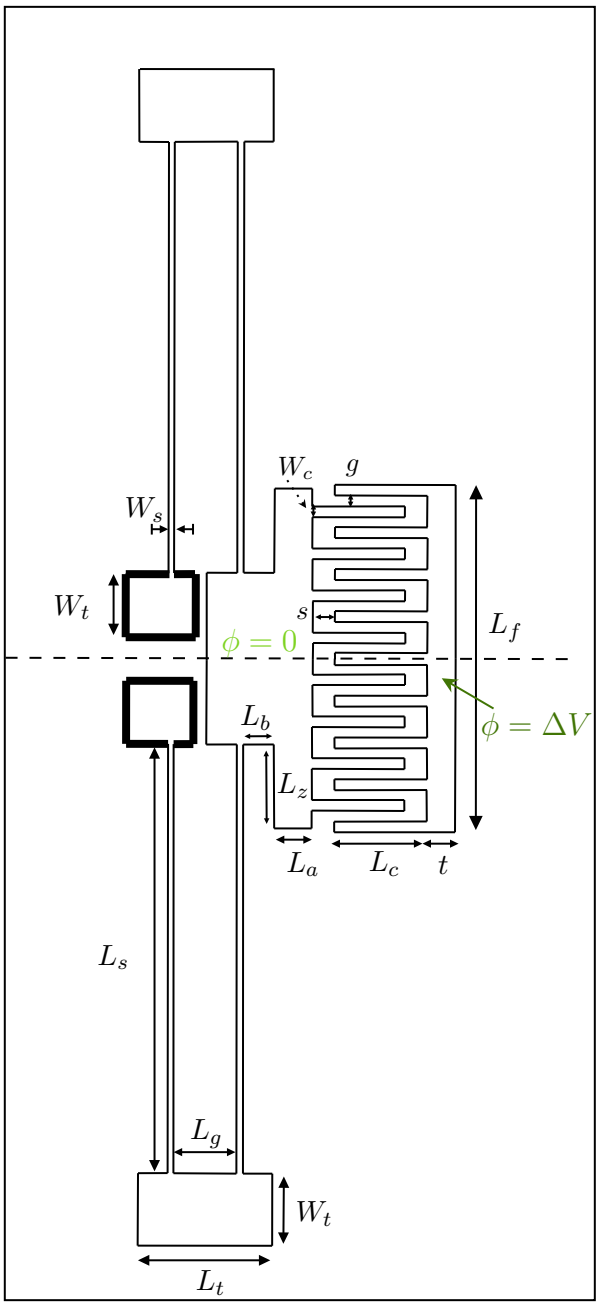


Figure 67: Geometrical description of the folded flexure actuator, implemented in the `FoldedFlexureBeam.geo` file. The right electrode is fixed and its electric potential is fixed to $\phi = \Delta V$. The whole flexure beam (left structure) is moving and is the mass of the electrical system ($\phi = 0$ [V]). The different dimensions are gathered in Table 9. An homogeneous Neumann condition (normal component of the electric field) is applied on the boundary of the remaining boundary of the domain. Air is located between the different electrodes: $\varepsilon = \varepsilon_0 = 8.85 \cdot 10^{-12}$ [F/m]. The dashed line represents the symmetry of the structure. The two anchors in bold are fixed, corresponding to homogeneous Dirichlet boundary conditions.

Dimension	W_t	L_t	L_g	L_s	L_b	L_z	L_a	L_c	t	L_f	W_c	g	W_s	s
[µm]	16	40	20	280	10	36	12	30	12	196	4	8	20	10

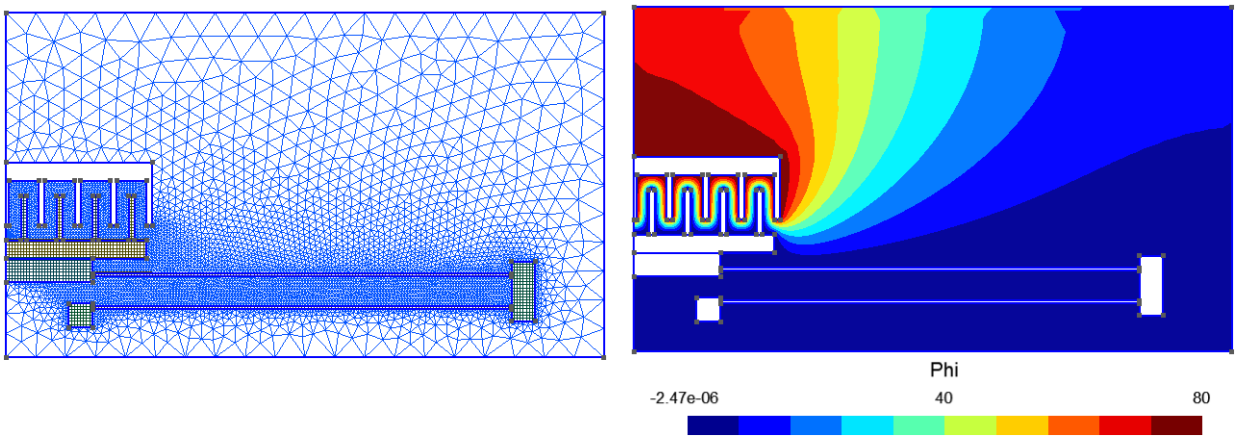
Table 9: Numerical value for the dimensions of the folded flexure beam introduced in Figure 67.

Figure 67 must be interpreted as rotated by 90 degrees. The folded flexure structure is made of silicon: $E = 160 \cdot 10^9$ [Pa], $\nu = 0.22$ [-], $\rho = 2300$ [kg/m^3]. The right fixed part corresponds in practice to the top electrode, with the moving structure below the fixed electrode. The bottom part, called the anchor, is fixed. The right part between the two horizontal beams of length L_s is called the truss, while the middle part below the different fins is called the base. The structure corresponds is the combination of the two first applications as it made of both clamped beams and comb-like system for the electrodes. Note that the dimensions of the structure are not the exact same dimensions as the ones introduced in [10]. Such a folded flexure beam is convenient because axial forces are reduced which facilitates large deflections.

6.3.1 Mesh and symmetry

In order to reduce the computation time, the symmetry between the left and right part of the structure is used and only the right part of the structure is studied. A peculiar attention must be paid to the boundary conditions prescribed on the surfaces represented by the dashed line in Figure 67. For the surfaces which are not in the mechanical moving structure, the normal component of the electric field must be equal to zero by symmetry, hence an homogeneous Neumann boundary condition is applied. For the part inside the moving structure of silicon, the horizontal displacement must also be equal to zero by symmetry and the corresponding boundary condition is prescribed in the `FoldedFlexureBeam.geo` file.

The initial structure of the mesh used for numerical simulations is represented in Figure 68(a). As can be observed, many elements are automatically generated between the two horizontal beams. The CPU time is increased by a lot as many boundary elements are located on the bottom half of the moving structure where the electric potential field is known to be equal to zero, as represented in Figure 68(b).



(a) (b)

Figure 68: (a) Initial global mesh structure for the folded flexure beam and (b) corresponding electric potential field ϕ [V].

As the electric field (opposed to the gradient of the electric potential) is equal to zero at the right of the truss, the BEM domain can be reduced to increase the rapidity of the numerical computation. The final mesh structure is represented in Figure 69. The mesh at the surface of the upper fixed electrode is automatically adapted to the density of the elements used in the FEM domain. Note again that the moving structure corresponding to the FEM domain has been divided into sub domains to allow a structured meshing using square first order elements to increase the accuracy of the computation. The results presented in this section are also validated using an unstructured mesh.

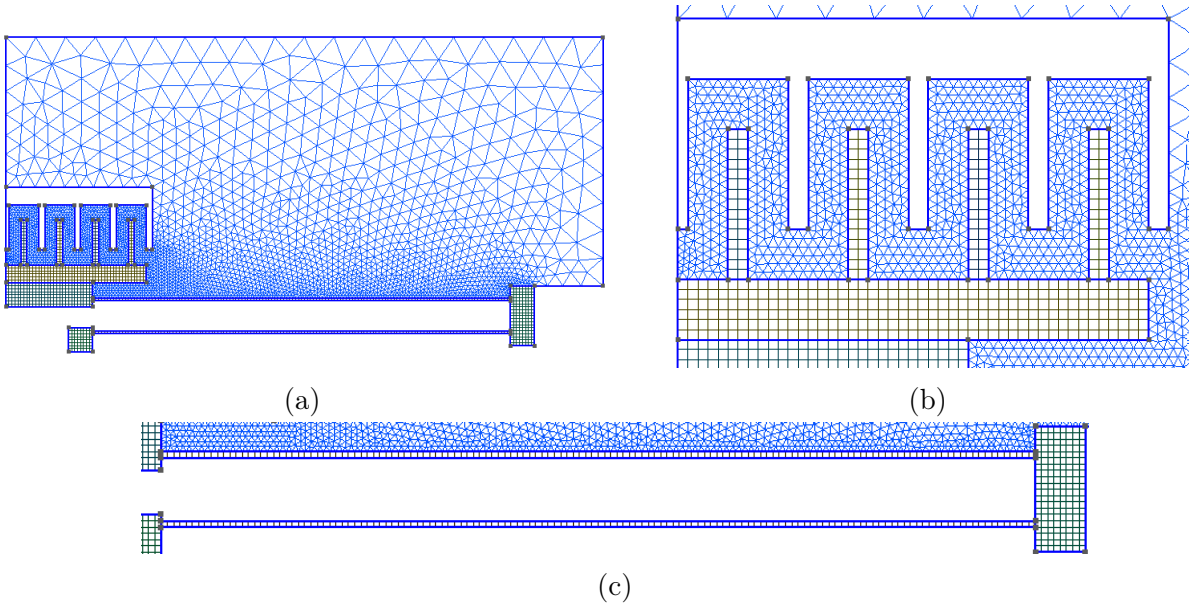


Figure 69: (a) Final global mesh structure for the folded flexure beam, (b) zoomed on the space between the electrodes and (c) zoomed on the horizontal beams.

6.3.2 Convergence of the maximal vertical displacement

To estimate the number of elements required in the FEM domain to obtain an accurate numerical solution, the convergence of the maximal vertical displacement for $\Delta V = 80$ [V] is studied when the mesh is progressively refined. The results are gathered in Table 10. As can be observed, an important number of elements is required for the vertical displacement to converge.

n_{FEM} [-]	N_{BEM} [-]	Maximal vertical displacement u_y [µm]
1582	583	0.746
5439	1109	1.067
19981	2160	1.225
43627	3209	1.274
76377	4258	1.30
118231	5307	1.32
169189	6536	1.33

Table 10: Convergence study of the maximal vertical displacement for the folded flexure beam with $\Delta V = 80$ [V], using the non-linear iterative solver. The number of nodes n_{FEM} is the total number of nodes in the Ω_{FEM} domain, while the number of elements N_{BEM} is the total number of linear elements on the Ω_{BEM} domain boundary.

In the following section presenting the physical fields in and around the structure, $n_{\text{FEM}} = 76377$ nodes are used for the numerical computation. However, only $n_{\text{FEM}} = 19981$ elements are used for performing the numerical simulations required to establish the voltage-displacement curve as the computation time becomes important when using the iterative non-linear solver and as many simulations are required in order to obtain these physical curves.

Figure 70 shows the convergence of the relative displacement difference as the number of iteration increases for several differences of potential. As for the two other applications, one can clearly see the exponential behaviour of the convergence in the semi-log plot.

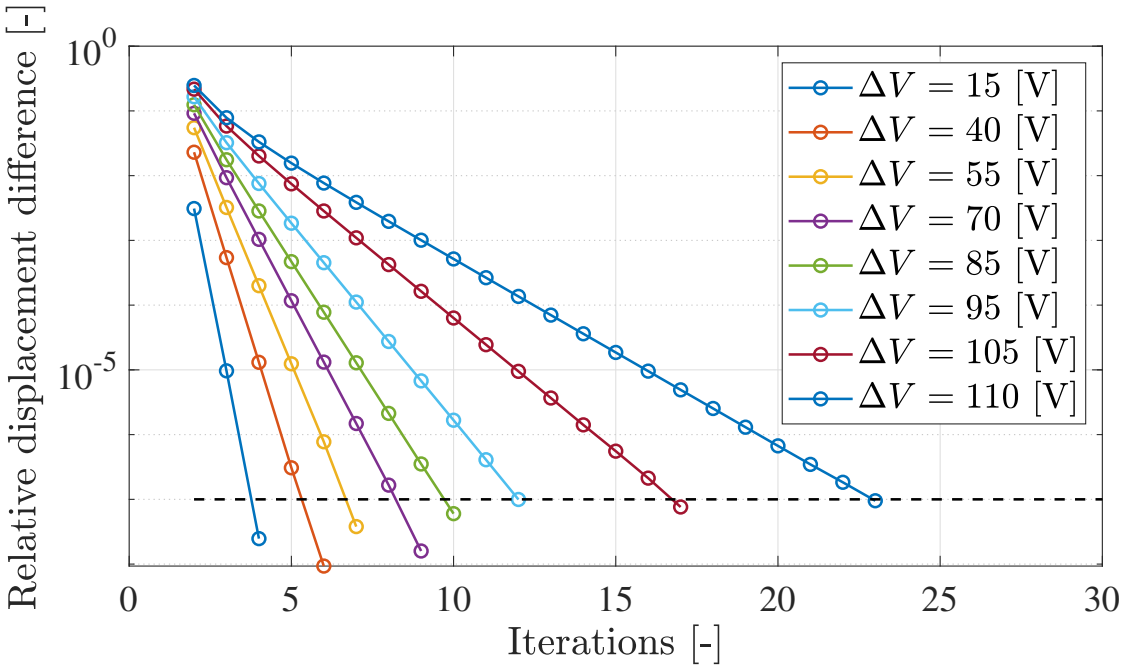


Figure 70: Evolution of the relative displacement difference as a function of the number of iterations of the iterative solver for the folded flexure beam in the configuration described in Figure 67.

6.3.3 Different physical fields

In this section, the fields obtained numerically are presented for $\Delta V = 110$ [V] and $n_{\text{FEM}} = 4258$ elements in the FEM domain using the non-linear iterative solver.

First, the deformed configuration of the structure is presented in Figure 71. As can be observed, the most important part of the vertical deformation takes place in the two horizontal beams connecting the anchor to the truss and connecting the truss to the anchor, which undergo bending. The homogeneous Dirichlet boundary condition on the displacement is respected at the anchor. There is almost no deformation in the truss (right part of the structure) and the displacement is uniform. Similarly, there is almost no deformation in the base and in the fins, where the displacement is vertical (the symmetry condition is respected) and the displacement is maximal.

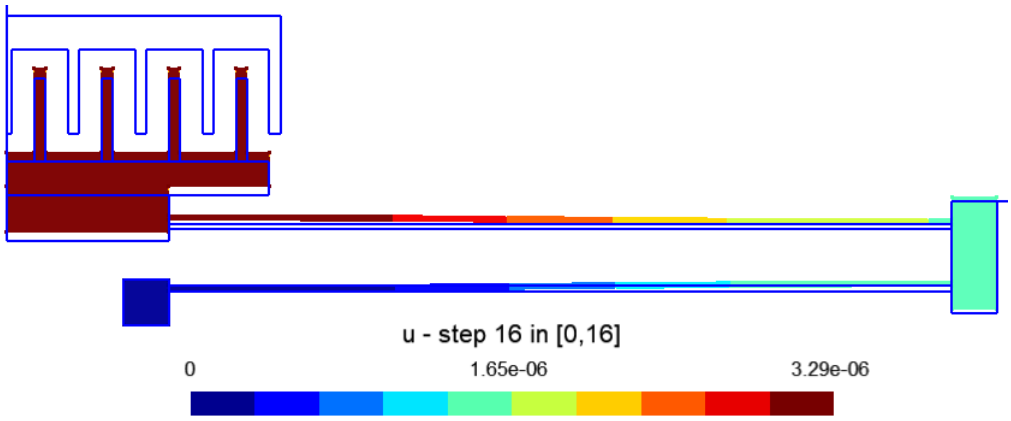
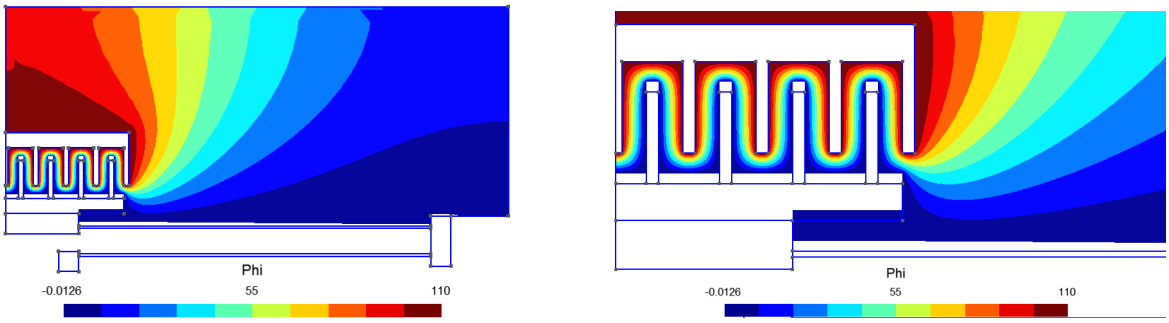


Figure 71: Deformed configuration and amplitude of the displacement $\|\boldsymbol{u}\|$ [m] of the folded flexure beam introduced in Figure 67 for $\Delta V = 110$ [V]. Results obtained with the two-way coupled non-linear solver, using 76377 nodes and square first order elements in the FEM domain Ω_{FEM}.

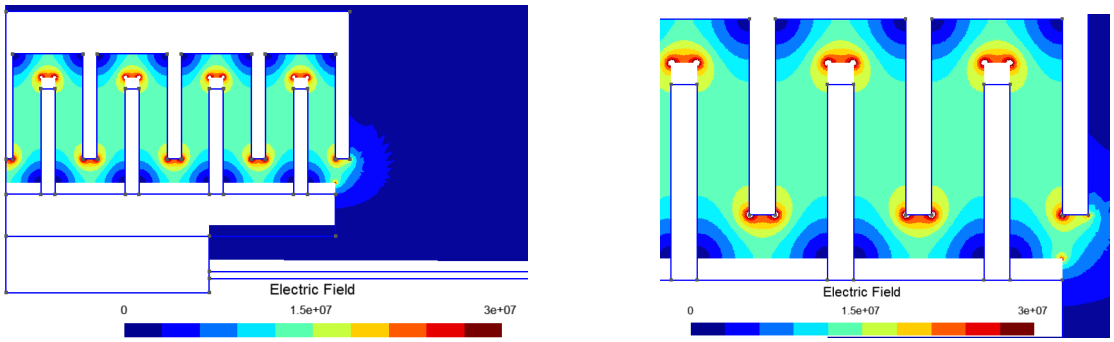


(a) (b)

Figure 72: (a) Global view and (b) zoomed on the free space between the electrodes view of the electric potential field ϕ [V] of the folded flexure beam introduced in Figure 67 for $\Delta V = 110$ [V]. Results obtained with the two-way coupled non-linear solver, using 4258 linear elements on the boundary of the BEM domain Ω_{BEM}.

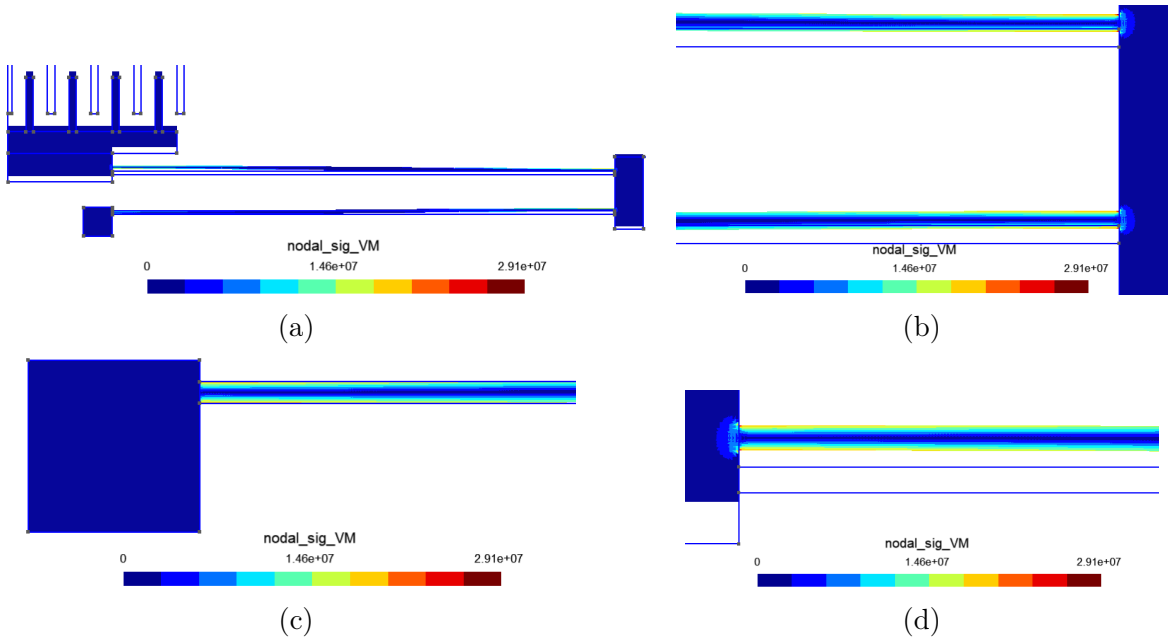
The electric potential field ϕ [V] is represented in Figure 72. When compared to the global form of the electric potential field obtained with the initial mesh in Figure 68(b), one can observe as the reduced mesh produces very similar results, which further justifies the use of such a reduced mesh. As can be seen in Figure 72(b), the electric potential between the electrodes is impacted by the deflection of the folded flexure beam. Between the fins of the moving part and the fins of the fixed electrode, the electric potential is very similar to the one observed in a plane capacitor and the equipotential lines are almost perfectly vertical. This is expected as it has already been observed for the comb-drive actuator in Figure 58(a).

The amplitude of the corresponding electric field is represented in Figure 73. Note that the direction of the electric field is opposed to the gradient of the electric potential field and is hence orthogonal to the equipotential lines observed in Figure 72(b). The electric field is almost uniformly equal to zero in the whole domain except in the free space between the electrodes. As mentioned above, the electric field between the fins is uniform and corresponds to the configuration of a plane capacitor. At the corner of the fins, the electric field seems to diverge as its amplitude reaches $\|\boldsymbol{E}\| = 5.65 \cdot 10^7$ [V/m]. Such a behaviour could be due to the sharp geometrical transition of the corner.



(a) (b)

Figure 73: Amplitude of the electric field $\|\boldsymbol{E}\|$ [V/m] for the folded flexure beam introduced in Figure 67 for $\Delta V = 110$ [V], (a) between and around the fixed and moving electrodes and (b) zoomed on the free space between the fins. Note that the maximal value of the color range has been reduced from $\|\boldsymbol{E}\| = 5.65 \cdot 10^7$ [V/m] to $\|\boldsymbol{E}\| = 3 \cdot 10^7$ [V/m] to ease the visualisation. Results obtained with the two-way coupled non-linear solver, using 4258 linear elements on the boundary of the BEM domain Ω_{BEM}.



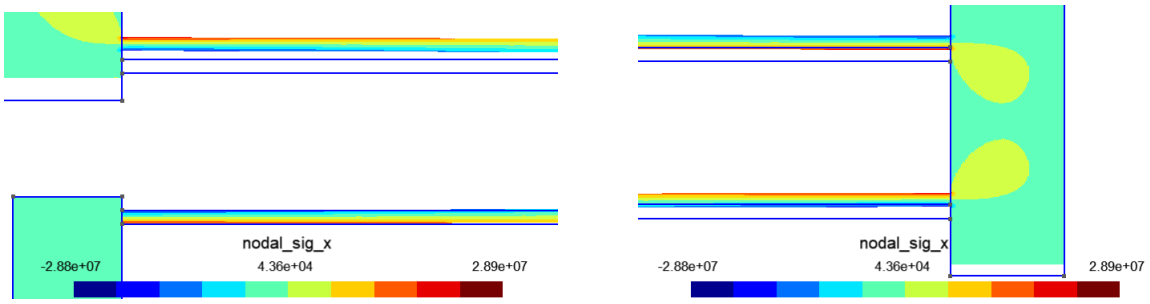
(a) (b)

(c) (d)

Figure 74: Equivalent von Mises stress field σ_{VM} [Pa] for the folded flexure beam introduced in Figure 67 for $\Delta V = 110$ [V], (a) in the global view, (b) zoomed on the truss, (c) zoomed on the anchor and (d) zoomed on the base. The results are displayed on the deformed configuration and are obtained with the two-way coupled non-linear solver, using 76377 nodes and square first order elements in the FEM domain Ω_{FEM}.

The corresponding von Mises stress field in the structure is represented in Figure 74. The stress is almost uniformly equal to zero in the anchor, in the truss and in the base. The stress is concentrated in the two horizontal beams and more particularly at the corresponding junctions with the bigger parts. From the shape of the deformed structure, one can guess that the stress field in the beams corresponds to a bending configuration. Indeed, one can observe in Figure 75 that the stress mainly corresponds to an axial σ_{xx} stress corresponding to the bending of the beams around the out-of-plane axis. Note that the maximal stress is

far lower than the initial yield stress of silicon $\sigma_y^0 = 165$ [MPa], so that the elastic assumption is still valid.



(a) (b)

Figure 75: Axial stress field σ_{xx} [Pa] for the folded flexure beam introduced in Figure 67 for $\Delta V = 110$ [V], (a) zoomed on the anchor and on the truss and (b) zoomed on the truss. The results are displayed on the deformed configuration and are obtained with the two-way coupled non-linear solver, using 76377 nodes and square first order elements in the FEM domain Ω_{FEM}.

6.3.4 Voltage-displacement curve

The most physical relation associated to the folded flexure beam is once again the vertical displacement of the base induced by the voltage difference ΔV applied between the electrodes. The numerical results are gathered in Figure 76, for both the linear and the non-linear iterative solver.

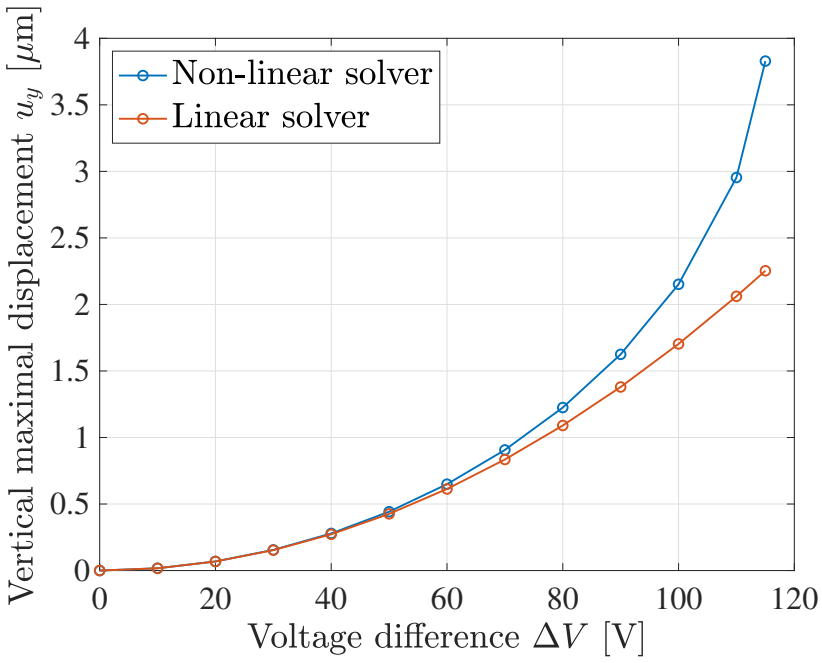


Figure 76: Evolution of the maximal vertical displacement u_y [µm] as a function of the voltage difference ΔV [V] applied on the fixed electrode of the folded flexure beam described in Figure 67. Results obtained with both the one-way coupled linear solver and the two-way coupled non-linear solver, using 19981 nodes and square first order elements in the FEM domain Ω_{FEM}.

As observed previously, the results are very similar for small voltage differences inducing small displacements. For the linear solver, the vertical deflection depends quadratically on the voltage difference, as explained in Section 6.1.4. For larger voltage differences, the iterative result is moving away from the quadratic curve and even diverges for $\Delta V_{\text{pi}} = 116$ [V], corresponding to the pull-in voltage of the structure. This shape of the voltage-displacement curve has already been encountered for the two first applications.

To further study the physical behaviour of the system, the length of the horizontal beams has been varied from $L_s = 160$ [µm] to $L_s = 280$ [µm] and the previous results have been reproduced. For the different simulations, the density of finite elements in the FEM domain has been kept constant.

The voltage-displacement curve is represented in Figure 77. As can be observed, the shape of the curve is rather general as it is quadratic for small displacements before diverging when reaching the pull-in voltage. For a given voltage difference, the vertical deflection is reduced when the length of the horizontal beams L_s is decreased. Hence, the pull-in voltage is greater for shorter beams as they are more rigid as discussed below. Note that the vertical deflection corresponding to the last stable configuration is of the same order of the magnitude no independently from the length of the horizontal beams.

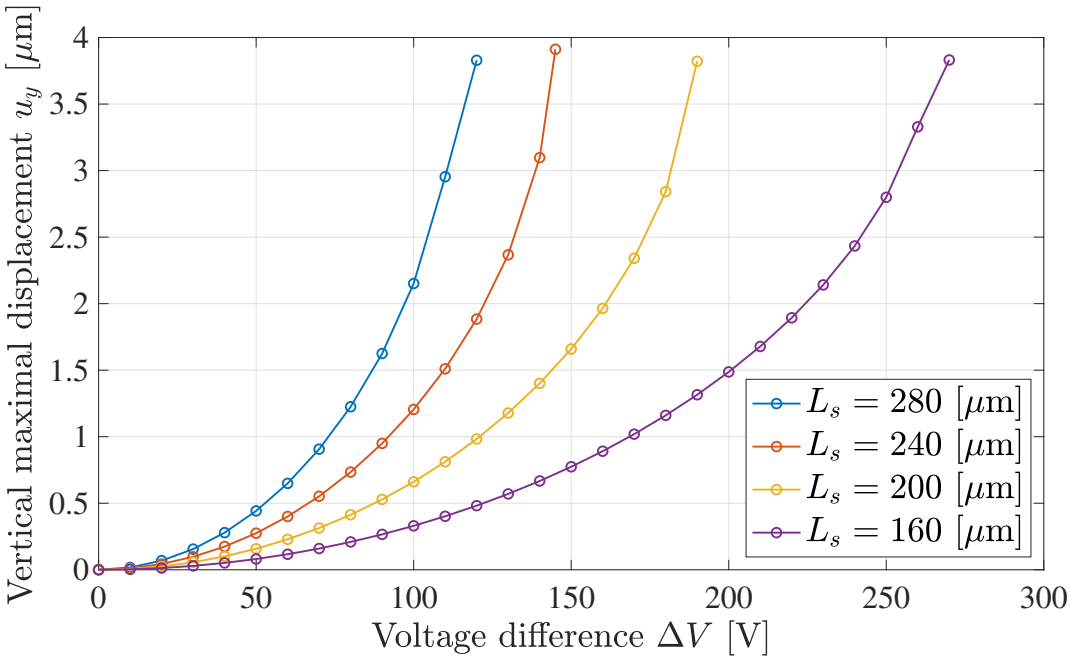
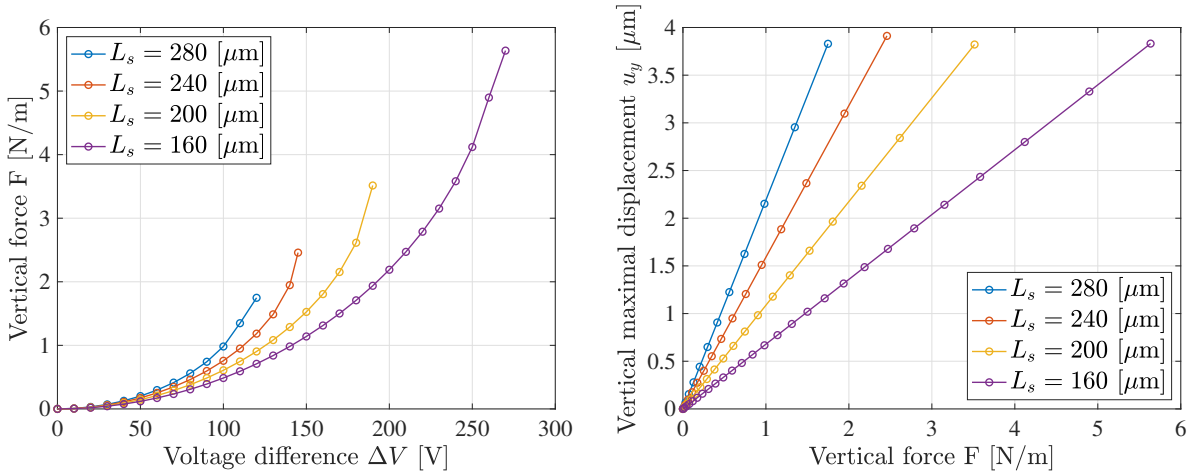


Figure 77: Evolution of the maximal vertical displacement u_y [µm] as a function of the voltage difference ΔV [V] applied on the fixed electrode of the folded flexure beam described in Figure 67 for different beam lengths L_s. Results obtained with the two-way coupled iterative solver.

The shape of the curves displayed just above can be explained by the physical coupling between the electrostatics and the kinematics of the structure, which are summarized in Figure 78.



(a) (b)

Figure 78: Evolution of (a) the vertical force F [N/m] applied on the folded flexure beam as a function of the voltage difference ΔV [V] applied between electrodes and (b) maximal vertical displacement [µm] as a function of the vertical force F [N/m] applied on the moving structure for different beam lengths L_s. Results obtained with the two-way coupled iterative solver.

The applied voltage difference induces an electrostatic vertical force on the moving structure, which is approximately a quadratic function of the voltage difference (at least for small displacements, as discussed in Section 6.1.4 and shown in Figure 65(a)). This electrostatic force induces a vertical deflection of the structure which once again impacts the distribution of the electric potential and the electric field and the electrostatic force. This mechanism shows how tightly the two physics are coupled. As can be observed in Figure 78(b), the vertical deflection is a linear function of the vertical force applied on the structure. Hence, it is convenient to introduce the structural stiffness k [N/m^2] of the folded flexure beam, defined as $F = k\ u_y$. Graphically, it corresponds to the inverse of the slope of the straight lines in Figure 78(b). As can be observed, the stiffness of the structure increases when the length L_s of the horizontal beams is decreased, as summarized in Table 11.

Beam length L_s [μm]	280	240	200	160
Structural stiffness k [10^6 N/m^2]	0.4565	0.6286	0.9197	1.4725

Table 11: Structural stiffness of the folded flexure beam described in Figure 67, computed numerically for different horizontal beam lengths L_s.

Note that, since the simulations have been carried out on half the geometry of a real folded flexure device, the obtained vertical force is half the true force. Therefore the structural stiffness presented here-above is also reduced by half in comparison to the real one.

6.4 Similarities between the different geometries

Although they exhibit very different geometrical configurations, several similarities have been found across the different studied applications. This shows that there are some phenomena that are truly related to the physical nature of the elasto-electrostatic interaction itself and not really related to the geometry of the problem.

The first phenomenon of interest is rather numerical and is the type of convergence of the coupled iterative solver. In fact, whatever the studied application (at least for the 3 cases presented here), it has been shown that the relative displacement difference between two iterations exponentially decreases as the number of iteration increases. This very fast convergence thus seems to be a characteristic of such coupled solver. This phenomenon has been observed for the three applications in Figures 52, 57 and 70, respectively.

Another very clear similarity is the relation binding the maximal displacement and the applied voltage. In fact, even if the exact quantities vary from one application to another, the general tendency is always the same: the maximal induced displacement is a quadratic function of the applied voltage for small displacements. This phenomenon has already been theoretically predicted in Section 6.1.4 and has now been shown numerically. Moreover, since the pull-in instability tends to make the structure diverge for a too high applied voltage, the quadratic curve always ends up being nearly vertical just before the pull-in. This corresponds, in fact to a situation where a small incremental applied voltage implies a huge displacement. Of course, this can only be observed with the non-linear solver since it implies large displacements.

Conclusion

Throughout the whole report, the implementation of the coupled FEM-BEM solver has been validated against many different geometries.

First, the separated solvers have been described theoretically and validated individually. Some specific features of the different methods have also been studied extensively, such as their order of convergence or their complexity. Note that the separated solvers also work independently, such that one can study purely elastic problems or purely electrostatic problems.

The one-way coupled solver, introducing the electrostatic pressure which is the basis for electrostatic actuation, has allowed to study the physics of MEMS devices in small displacements configuration. The corresponding numerical results are accurate as long as the displacement of the mechanical structure does not modify the distribution of the electric potential field around the structure. Under these conditions, it has been shown that the displacement of the moving structure is often a quadratic function of the voltage difference applied across the device.

To tackle more realistic problems and to allow the characterization of the so-called pull-in instability which involves large displacements, a non-linear FEM solver has been implemented in order to deal with large rotations of the local finite elements. Once again, the isolated solver has been validated against results retrieved from the literature. The corresponding Newton-Raphson algorithm has been found to be particularly efficient and robust.

The two-way coupled solver, in which both the BEM solver and the non-linear FEM solver communicate until an equilibrium configuration is reached, has allowed to fully couple the two physics involved in electrostatic actuation. While the electrostatics impact the mechanics through the electrostatic pressure, the mechanics impact the electrostatics through the displacement of the different electrodes.

The first application, corresponding to the electrostatic actuation of a clamped micro beam, has confirmed the accuracy of the implemented numerical solver, as the pull-in voltage of the device computed with the code is very similar to what has been found in the literature.

The second application has introduced the concept of comb-drive device, which is widely used for electrostatic actuation in microsystems. The physics of such systems have been studied extensively and many different characteristic curves have been obtained numerically. This second geometry has also allowed to study an accelerometer, converting a constant acceleration into a voltage difference. In the discussed configuration, the one-way coupled solver is accurate as the displacement of the moving structure is feedback-controlled to be equal to zero.

The final application is the folded flexure beam, which combines mechanisms encountered in the first applications, as the geometry is composed of both beams and a comb-drive device. The physical behaviour of the device, as well as the rigidity of the structure depending on the beam length, have also been quantified.

For the different applications encountered in the report, the voltage-displacement curves are very similar, which highlights the generality of the physical principle behind the pull-in mechanism. Moreover, the convergence of the iterative non-linear solver is also equivalent for different geometries, which shows that it is not specific to a particular configuration.

It is important to notice that the present implementation of the coupled solver could be improved. First, it could be generalized to three-dimensional problems. However, even if the generalization of the FEM code would be straightforward, adapting the BEM code would be quite more complex.
Also, one could implement the resolution of the dynamical behaviour of the system by introducing inertial terms in the finite element formulation. Such an improvement would require to introduce a time-integrating numerical algorithm.
The most useful upgrade of the code would be the possibility to handle composite materials. In the FEM domain, it could be implemented by assigning different material properties to different parts of the structure. In the BEM domain, the Laplace equation is not valid across the interface between different materials (except if the dielectric permittivity is the same in the two regions) and the whole numerical discretization of the problem should be adapted to take this physical discontinuity into account.

References

[1] J.P. Ponthot, *MECA0036: Finite Element Method*, ULiège, personal notes, 2020-2021.

[2] Eigen documentation, *Solving sparse linear systems*, consulted on May 13, 2022.

[3] J. Katsikadelis, *The Boundary Element Method for Engineers and Scientists. Theory and application.*, 2016.

[4] J.-P. Ponthot, *MECA0446: Continuum Mechanics*, ULiège, personal notes, 2021-2022.

[5] J.-M. Battini, *A non-linear corotational 4-node plane element*, Mechanics Research Communications, Department of Mechanics, KTH, Royal Institute of Technology, 2008.

[6] N. Sonnenberg et al., *Stiction in surface micromachining*, Journal of Micromechanical Engineering, University of Twente, August 1996.

[7] P.M. Osterberg et al., *A quantitative model for the measurement of residual stress using electrostatic pull-in of beams*, Proc. Solid-State Sensor and Actuator Workshop, 1994.

[8] M. Younis, *MEMS Linear and Nonlinear Statics and Dynamics*, Springer, 2011.

[9] J.-P. Jaspart, *MECA0001: Mécanique des Matériaux*, ULiège, personal notes, 2019-2020.

[10] S. Gupta et al., *Optimizing the Performance of MEMS Electrostatic Comb Drive Actuator with Different Flexure Springs*, Electronic Science Department, Kurukshetra University, 2012.