POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Transient simulation of charge accumulation in HVDC cables during switching

Advanced Programming for Scientific Computing

Project Report

Author: **Alessandro Lombardi**

Student ID: 946444
Supervisor: Prof. Carlo De Falco
Academic Year: 2021-22

Contents

Introduction

High Voltage Direct Current (HVDC) is an electric power transmission system commonly used to transmit power over long distances and for undersea cables, making it ideal for offshore windfarms. Other uses include connecting unsynchronized AC grids to each other, such as the ones of neighbouring countries. It requires a converter-transformer at each end in order to connect it to the local AC grid, however, due to the absence of skin effect transmission, losses are 50% less than AC lines at the same voltage. Moreover HVDC requires less conductor per unit distance than an AC line since there is no need to support three phases.
The standard structure of the cable for HDVC consist of a coaxial cable surrounded by an insulation layer made by high-pressure oil or of mass impregnated paper with high-viscosity insulating compound.[2] In both these cases, any damage to the cable which causes leakage of the fluid will need expensive repairs before being able to be used again. In more recent years, polymeric materials have been used as insulators in HVDC applications. These so-called extruded cables have the advantage of being smaller and lighter, moreover they are not subjects to leakage and don't require additional equipment to maintain the fluid pressure. A disadvantage of these insulating materials is that the insulation of such cables exhibit complex non-linear conduction current (charge transport in dielectric) when used under high DC stress [7]. The conduction current is also considered responsible for the space charge accumulation, which is claimed to be the main factor accelerating degradation of polymeric insulation in HVDC cables with respect to HVAC [4]. This is the reason why understanding the dominant chemical and physical processes in the insulation, achieved through experiments, development of theoretical models and numerical simulations, can help to build reliable systems.
The goal of this project is to study the electric charge distribution in HVDC insulation materials in non-stationary conditions. We first develop a relatively simple numerical model for evaluating the behavior of the electric field and the space charge inside the insulator. Then we implement it using the bim++ library.

This report is structured as follows. In Chapter 1 the mathematical model is derived and discussed. Chapter 2 focuses on the coding aspect of the project. Finally in Chapter 3 we present the numerical results.

1 | Mathematical Model

1.1. Derivation of the equations

The following mathematical model will be derived under the quasi-electrostatic approximation:

$$\nabla \times \boldsymbol{E} = 0 \tag{1.1}$$

where $\boldsymbol{E}$ is the electric field. This allows us to introduce a scalar potential φ such that

$$\boldsymbol{E} = -\nabla\varphi \tag{1.2}$$

Under the same assumption, let's now consider Maxwell's equations:

$$\nabla \cdot \boldsymbol{D} = \rho, \tag{1.3a}$$

$$\nabla \times \boldsymbol{H} - \frac{\partial \boldsymbol{D}}{\partial t} = \boldsymbol{J}. \tag{1.3b}$$

Equation 1.3a, also known as Gauss' law, relates the dielectric displacement vector $\boldsymbol{D}$ to the net density of charge per unit volume ρ.
Inside a dielectric material the displacement vector is given by

$$\boldsymbol{D} = \varepsilon_0 \boldsymbol{E} + \boldsymbol{P} \tag{1.4}$$

where ε_0 is the vacuum permittivity and $\boldsymbol{P}$ is the polarization vector which we can assume to be due to a number of different processes, each behaving according to the Debye relaxation model [5]

$$\boldsymbol{P} = \boldsymbol{P}_\infty + \sum_k \boldsymbol{P}_k \tag{1.5}$$

where

$$\boldsymbol{P}_\infty = \varepsilon_0 \chi_\infty \boldsymbol{E} \tag{1.6}$$

and each of the contributions $\boldsymbol{P}_k$ obeys a relaxation–type ordinary differential equation of the form

$$\tau_k \frac{\partial}{\partial t} \boldsymbol{P}_k = \varepsilon_0 \chi_k \boldsymbol{E} - \boldsymbol{P}_k \tag{1.7}$$

where τ_k is the corresponding time constant and the parameter χ_k represent the electric susceptibility for a given frequency component k.
It has been shown that this model for the polarization can correctly represent currents in

impregnated-paper insulation during transients regimes [3]. However, it becomes much less accurate when applied to polymeric materials such as polypropylene. The different behaviour observed in polymeric material can be ascribed to the the presence of charge carrier traps in such materials.
To account for this phenomenon, let model the charge density ρ as

$$\rho = q(p - n + b) \tag{1.8}$$

where p are the (positively charged) mobile charges, b are the (positively charged) trapped charges and n the (negatively) charged trap states. We assume the trap states to be fixed

$$\frac{\partial n}{\partial t} = 0 \tag{1.9}$$

while the model for charge trapping will be discussed later

$$\frac{\partial b}{\partial t} = \dot{b}(\boldsymbol{J}, b) \tag{1.10}$$

Let's now consider equation 1.3b, in which $\boldsymbol{H}$ is the magnetic field strength and $\boldsymbol{J}$ is the current density. $\boldsymbol{J}$ can be expressed as a function of the electric field:

$$\boldsymbol{J} = \sigma \boldsymbol{E} \tag{1.11}$$

where σ is the electrical conductivity which is a function of the electric field and the temperature:

$$\sigma = (1 - a)\sigma_1(|\nabla\varphi|, T) + a\sigma_0(T) \tag{1.12}$$

with

$$\tau_a \frac{\partial a}{\partial t} = \bar{a}(|\nabla\varphi|) - a \tag{1.13}$$

$$\bar{a} = \begin{cases} 0, & |\nabla\varphi| > e_{tr} \\ 1, & |\nabla\varphi| \leq e_{tr} \end{cases} \tag{1.14}$$

$$\frac{1}{\tau_a} = \begin{cases} \frac{\kappa+1}{c} \sqrt[\kappa+1]{a(t)}, & |\nabla\varphi| > e_{tr} \\ \frac{1}{\bar{\tau}_a}, & |\nabla\varphi| \leq e_{tr} \end{cases} \tag{1.15}$$

where τ_a is the time constant and $e_{tr},\ \kappa,\ c$ and $\bar{\tau}_a$ are constants.
By combining 1.2, 1.3a and 1.4 we get

$$\rho = \nabla \cdot \boldsymbol{D} = -\nabla \cdot (\varepsilon_0 \nabla\varphi) + \nabla \cdot \boldsymbol{P} = -\nabla \cdot (\varepsilon_0(1 + \xi_\infty)\nabla\varphi) + \sum_k \nabla \cdot \boldsymbol{P}_k \tag{1.16}$$

Using 1.8 and $\varepsilon_\infty = 1 + \chi_\infty$, 1.16 becomes

$$-\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla\varphi) = q(p - n - b) - \sum_k \nabla \cdot \boldsymbol{P}_k \tag{1.17}$$

If we now take the divergence of 1.8 and set $\pi_k = \nabla \cdot \boldsymbol{p}_k$ we get

$$\tau_i \frac{\partial}{\partial t}\pi_i = \frac{\chi_i}{\varepsilon_\infty}\left(q(p - n - b) - \sum_k \pi_k\right) - \pi_i \tag{1.18}$$

Applying now the divergence operator to 1.3b and using 1.11, 1.3a, 1.8 we get:

$$0 = \nabla \cdot (\nabla \times \boldsymbol{H}) = \nabla \cdot \boldsymbol{J} + \frac{\partial}{\partial t} \nabla \cdot \boldsymbol{D} = -\nabla \cdot (\sigma \nabla \varphi) + q \frac{\partial}{\partial t}(p - n - b) \tag{1.19}$$

and using 1.9 and 1.10 we get

$$\frac{\partial n}{\partial t} + \nabla \cdot \left(\frac{\sigma}{q} \nabla \varphi \right) = -\dot{b}(\boldsymbol{j}, b) \tag{1.20}$$

If we now assume that $n \to 0$, where n is the number of charged trap states, from 1.17, 1.19, 1.18 and 1.13 we get the Level 2 model:

$$\begin{cases} -\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla \varphi) = \rho - \sum_k \pi_k \\ \frac{\partial \rho}{\partial t} + \nabla \cdot (\sigma \nabla \varphi) = 0 \\ \tau_i \frac{\partial}{\partial t} \pi_i = \frac{\chi_i}{\varepsilon_\infty} (\rho - \sum_k \pi_k) - \pi_i \\ \tau_a \frac{\partial a}{\partial t} = \bar{a}(|\nabla \varphi|) - a \end{cases} \tag{1.21}$$

With the further assumption $a = 0$, 1.21 simplifies to the Level 1 model

$$\begin{cases} -\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla \varphi) = \rho - \sum_k \pi_k \\ \frac{\partial \rho}{\partial t} + \nabla \cdot (\sigma \nabla \varphi) = 0 \\ \tau_i \frac{\partial}{\partial t} \pi_i = \frac{\chi_i}{\varepsilon_\infty} (\rho - \sum_k \pi_k) - \pi_i \end{cases} \tag{1.22}$$

Finally, by assuming also that $\chi_k = 0 \ \forall k$ we get the Level 0 model

$$\begin{cases} -\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla \varphi) = \rho \\ \frac{\partial \rho}{\partial t} + \nabla \cdot (\sigma \nabla \varphi) = 0 \end{cases} \tag{1.23}$$

1.2. Boundary conditions

We now consider Ω to be a cubic domain of size L. Since the insulator is in contact with two conductors at different voltages, for the electric potential φ we impose on the top face homogeneous Dirichlet boundary conditions, whereas on the bottom phase we impose a potential of $\varphi = V_0$. Since the initial conditions are $\varphi = 0$, we impose a time-dependent boundary condition such that it is $\varphi = 0$ at $t = 0$:

$$\varphi(L, t) = V_0 \left(1 - e^{-t/\tau}\right) \tag{1.24}$$

where τ is a time constant such that $\tau << T$. On the other faces we have Neumann boundary conditions:

$$-\varepsilon \nabla \varphi \cdot \boldsymbol{n} = \rho \quad \text{on } S_L \tag{1.25}$$

For the variable ρ we don't have to impose boundary conditions since in the equations it doesn't appear under a spatial differential operator.

1.3. Time discretization

Let's now consider the Level 0 model (1.23): it consists of a system of two equations, a reaction-diffusion equation and a continuity equation, in two unknowns, ρ and φ. We will the interval $[0, T]$ as time domain for our model.
We then divide it in K intervals of size Δt such that $K\Delta t = T$, and we denote by $(\rho_k, \varphi_k)_{k=0}^{K}$ the solutions at each time step. To perform the temporal integration, we use a Implicit Euler (IE) scheme:

$$\begin{cases} -\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla \varphi_{k+1}) - \rho_{k+1} = 0 & k = 0, \dots, K-1 \\ \rho_{k+1} - \nabla \cdot (\Delta t \sigma \nabla \varphi_{k+1}) = \rho_k & k = 0, \dots, K-1 \\ \rho_0 = 0 \\ \varphi_0 = 0 \end{cases} \tag{1.26}$$

where we consider 0 as initial conditions for both ρ and φ. Note that the charge density appears as a reaction term in both equations (ρ_{k+1}) and as a known term in the second equation (ρ_k).

1.4. Spatial discretization: Octree grid

Since we are interested in an application of our model to a cable insulator, the physical domain Ω we will consider will be 3-dimensional. The spacial discretization is handled by the bim++ library using an Octree data structure. Octree is a tree data structure in which each internal node has exactly eight children, and it is used to partition a three-dimensional space by recursively subdividing it into eight octants or regions. An Octree mesh is generated starting from a squared structure and iteratively refining or coarsening its elements in a non conforming way (Figure 1.1).

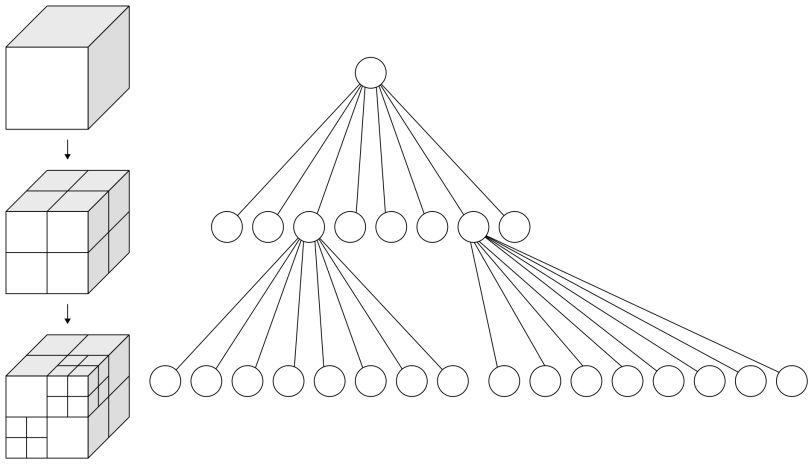

Figure 1.1: Octree mesh with level 1 global refinement and level 2 refinement in element2 and 6.

In particular, if we refer to the root of the tree as the element at level 0, we can say that a

subelement has level N if it has been obtained by refining the root N times. Analogously, it is possible to coarsen the mesh, aggregating eight octants in order to go back to have only their parent. To every octant we can associate an index, computed following the z-ordering (shown in Figure 1.2 for a Quadtree), which must be updated after any refinement. To every node of each quadrant are associated two indices: a local index, internal to the quadrant itself, and a global index, related to the position of the node in the full mesh.

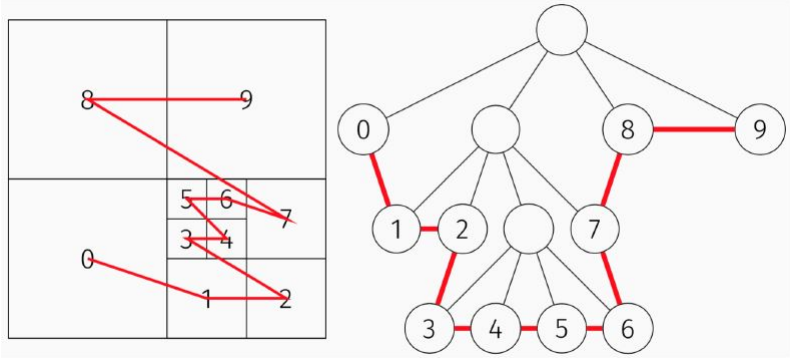

Figure 1.2: Example of the global numbering for a Quadtree mesh.

Each cubic element of the mesh is characterized by the same numbering of faces, edges and nodes, which is the one reported in Figure 1.3.

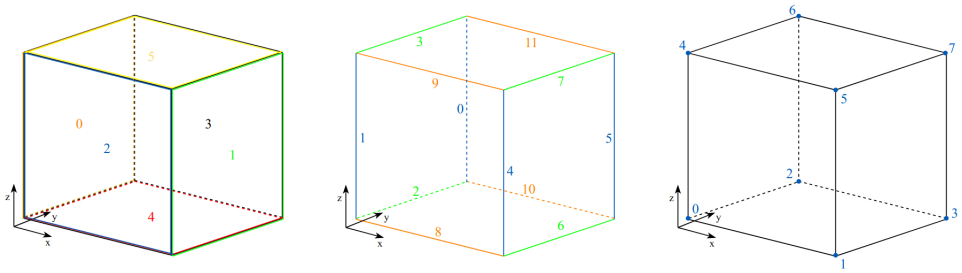

Figure 1.3: From left to right: numbering of faces, edges, nodes of each quadrant.

The refinement and coarsening processes can lead to hanging nodes, which are nodes lying on an edge which is refined for two elements, but not for the neighboring one (Figure 1.4). Hanging nodes have the usual local numbering but they do not have a global numbering, instead they are related to the global numbering of their parents, i.e. to the global numbering of the two non-hanging nodes lying on the same edge.

An Octree mesh is a partition τ_h of Ω in N_h elements such that:

$$\tau_h = \{\Omega^k\}_{k=1}^{N}, \qquad \Omega_h = \bigcup_{k=1}^{N} \Omega^k \subseteq \Omega \tag{1.27}$$

We can now define a space of continuous piecewise tri-linear polynomials $Q_h^1(\tau_h)$ over the given partition τ_h:

$$Q_h^1(\tau_h) = \left\{ u \in C^0(\bar{\Omega}_h) \text{ s.t. } u|_{\Omega^k} \in \mathbb{P}_{1,1,1}(\Omega^k) \ \forall \Omega^k \in \tau_h \right\} \tag{1.28}$$

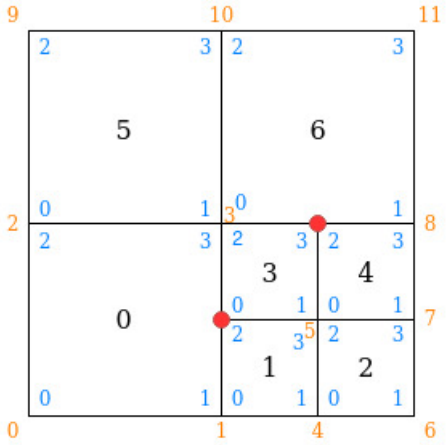

Figure 1.4: Example of hanging nodes for Quadtree mesh. Hanging nodes are marked with red spots, black numbers refer to quadrant numbering, blue to node local indices, orange to node global indices.

where

$$\mathbb{P}_{m,n,l}(\Omega) = \{p : \Omega \to \mathbb{R} \text{ s.t. } p(x,y,z) = \sum_{i \le m,\ k \le n,\ k \le l} a_{ijk} x^i y^j z^k \ \forall (x,y,z) \in \Omega\} \tag{1.29}$$

Since the Octree mesh can be non-conforming, the space $Q_h^1(\tau_h)$ defined in 1.28 is not the finite element space used for our solution; however the actual space $\tilde{Q}_h^1(\tilde{\tau}_h)$ is a subspace of $\bar{Q}_h^1(\bar{\tau}_h)$ if we consider $\bar{\tau}_h$ to be the mesh obtained with uniform refinement with level equal to the maximum level used in $\tilde{\tau}_h$. $\tilde{Q}_h^1(\tilde{\tau}_h)$ is obtained by modifying the starting space at each refinement and coarsening step.
In order to describe the refining procedure, consider the situation where we take an octant form refinement level N to $N+1$. Let τ_h be the original mesh, $\tilde{\tau}_h$ the one obtained from the refinement and let $\bar{\tau}_h$ br the one obtained from an uniform refinement of level $N+1$. Now let $\{\varphi_i\} \subset Q_h^1(\tau_h)$, $\{\tilde{\varphi}_i\} \subset \tilde{Q}_h^1(\tilde{\tau}_h)$ and $\{\bar{\varphi}_i\} \subset \bar{Q}_h^1(\bar{\tau}_h)$ we the corresponding basis. Then, if $\varphi_k, \dots, \varphi_{k+7} \in Q_h^1(\tau_h)$ are the basis elements for the vertex of the considered element and $\bar{\varphi}_{k+8} \in \bar{Q}_h^1(\bar{\tau}_h)$ is the basis element for the newly created vertex at the centre of the element, then we can define

$$\tilde{\varphi}_{k+i} = \varphi_{k+i} - \frac{1}{8}\bar{\varphi}_{k+8} \quad \forall i = 0, \dots, 7 \tag{1.30}$$

The other basis elements of $\tilde{Q}_h^1(\tilde{\tau}_h)$ are the same of $Q_h^1(\tau_h)$, plus the addition of $\bar{\varphi}_{k+8}$. Note that $\tilde{Q}_h^1(\tilde{\tau}_h) \subset \bar{Q}_h^1(\bar{\tau}_h)$. Moreover, its basis function still satisfy:

$$\sum_i \tilde{\varphi}_i = 1, \qquad \tilde{\varphi}_i(x_j) = \delta_{ij} \quad \forall x_j \in \tilde{X} \tag{1.31}$$

where $\tilde{X}$ is the set of nodes defining $\tilde{\tau}_h$.

Coarsening is performed in a similar fashion, by adding a fraction of the deleted basis φ_k function to the surround ones in order to maintain the partition of unity property. If $\tilde{\varphi}_i$

are the resulting basis functions defined on the vertices of the octant after the coarsening, we have:

$$\tilde{\varphi}_i = \varphi_i + \frac{1}{2}\sum_{j \in M_i^B} \varphi_j + \frac{1}{4}\sum_{j \in M_i^F} \varphi_j + \frac{1}{8}\varphi_k \tag{1.32}$$

where M_i^B is the set of nodes that share an edge with node k and M_i^F is the set of nodes that share a face with node k.

2 | bim++ library

The bim++ library is an interface to p4est (and its 3D version p8est), the dynamic backend manager of quadtree (octree). It is an efficient, parallel, scalable finite element code that allows to solve advection-diffusion-reaction problem.

2.1. Quadtree/Octree

The first step is to define the mesh object using the class tmesh_3d.

```
tmesh_3d tmsh;
tmsh.read_connectivity (simple_conn_p, simple_conn_num_vertices,
                        simple_conn_t, simple_conn_num_trees);
```

We then import the connectivity, which consists of the nodes and coordinates of the vertex of our cube.

```
constexpr p4est_topidx_t simple_conn_num_vertices = 8;
constexpr p4est_topidx_t simple_conn_num_trees = 1;
const double simple_conn_p[simple_conn_num_vertices*3] =
  {0., 0., 0., 0.001, 0., 0., 0., 0.001, 0., 0.001, 0.001, 0.,
   0., 0., 0.001, 0.001, 0., 0.001, 0., 0.001, 0.001, 0.001, 0.001,
   0.001};
const p4est_topidx_t simple_conn_t[simple_conn_num_trees*9] =
  {1, 2, 3, 4, 5, 6, 7, 8, 1};
```

Here is an example of refinement function, which assign a value greater that zero to each octant which has to be refined, zero otherwise.

```
static int
ball_refinement (tmesh_3d::quadrant_iterator quadrant)
{
  int currentlevel = static_cast<int> (quadrant->the_quadrant->level);
  double xcoord, ycoord, zcoord, dist = .0;
  int retval = 0;
  for (int ii = 0; ii < 8; ++ii)
    {
      xcoord = quadrant->p(0, ii);
      ycoord = quadrant->p(1, ii);
      zcoord = quadrant->p(2, ii);

      dist = std::sqrt (std::pow (xcoord - b_x, 2) +
                        std::pow (ycoord - b_y, 2) +
                        std::pow (zcoord - b_z, 2));
      if (dist < 1.1 * b_r)
```

```
        {
          retval = maxlevel_ball - currentlevel;
          break;
        }
    }
  if (currentlevel >= maxlevel_ball)
    retval = 0;

  return (retval);
}
```

We finally perform uniform and non-uniform refinement

```
int recursive = 1;
tmsh.set_refine_marker (uniform_refinement);
tmsh.refine (recursive);

tmsh.set_refine_marker(ball_refinement);
tmsh.refine (recursive);
```

2.2. Assembly

We first define the orderings, which allow us to differentiate between the two unknowns φ and ρ is the solution vector. They also identify which rows of the system matrix correspond to which equation in our linear system.

```
ord0 = [] (tmesh_3d::idx_t gt) -> size_t { return dof_ordering<2, 0> (gt
    ); },
ord1 = [] (tmesh_3d::idx_t gt) -> size_t { return dof_ordering<2, 1> (gt
    ); };
```

The following lines assemble in the matrix "A" the part related to the diffusion of φ: $-\nabla \cdot (\varepsilon_0 \varepsilon_\infty \nabla \varphi)$ for the first equation and $-\nabla \cdot (\Delta t \sigma \nabla \varphi)$ for the second.

```
// advection_diffusion
bim3a_advection_diffusion (tmsh, epsilon, zero, A, true, ord0, ord0);
bim3a_advection_diffusion (tmsh, sigma, zero, A, true, ord1, ord0);
```

These lines assemble in the matrix "A" the part related to the reaction term $-\rho$ in the first equation and ρ in the second.

```
// reaction
bim3a_reaction (tmsh, delta0, zeta0, A, ord0, ord1);
bim3a_reaction (tmsh, delta1, zeta1, A, ord1, ord1);
```

The homogeneues and non-homogeneus time-dependent boundary conditions are set up and implemented for the variable φ.

```
dirichlet_bcs3 bcs0;

bcs0.push_back (std::make_tuple (0, 4, [](double x, double y, double z){
    return 0.0;})); //bottom
bcs0.push_back (std::make_tuple (0, 5, [time](double x, double y, double
    z){return 1.5e4 * (1 - exp(-tau/10.0));})); //top
```

```

bim3a_dirichlet_bc_bis (tmsh, bcs0, A, sol, ord1, ord0, false);
```

Since this was the first case of system of differential equation in three dimensions in bim++ with dirichlet boundary conditions on one variable only, the following functions needed to be implemented.

```
template <class T>
void
bim3a_dirichlet_bc (tmesh_3d& mesh, const dirichlet_bcs3& bcs,
                    sparse_matrix& A, T& rhs,
                    const ordering& ordr,
                    const ordering& ordc,
                    const bool& only_rhs)
{
  int boundary_idx, tree_idx;
  unsigned int row, col;

  std::set<unsigned int> marked;

  double value;

  for (auto quadrant = mesh.begin_quadrant_sweep ();
       quadrant != mesh.end_quadrant_sweep ();
       ++quadrant)
    {
      tree_idx = quadrant->get_tree_idx ();

      for (int i = 0; i < 8; ++i)
        {
          boundary_idx = quadrant->e (i);
          row = ordr (quadrant->gt (i));
          col = ordc (quadrant->gt (i));

          // If current node is on boundary and has not
          // been handled before.
          if (boundary_idx != tmesh_3d::quadrant_t::NOT_ON_BOUNDARY
              && marked.count(row) == 0)
            {
              // Loop over all the boundary conditions.
              for (size_t bc = 0; bc < bcs.size (); ++bc)
                // If this boundary condition matches with
                // the current node.
                if (std::get<0> (bcs[bc]) == tree_idx
                    && std::get<1> (bcs[bc]) == boundary_idx)
                  {
                    // Mark current node so to avoid duplicate
   operations.
                    marked.insert (row);

                    // Evaluate bc at current node
                    value = (std::get<2> (bcs[bc]))
                            (quadrant->p (0, i),
                              quadrant->p (1, i),
                              quadrant->p (2, i));
```

```
48
49                      bim3a_dirichlet_bc_loc (A, rhs, row, col, value,
    only_rhs);
50                    }
51                }
52            }
53        }
54    }
```

```
1  template <class T>
2  void
3  bim3a_dirichlet_bc_loc (sparse_matrix& A,
4                          T& rhs,
5                          const unsigned int& row,
6                          const unsigned int& col,
7                          const double& value,
8                          const bool& only_rhs)
9  {
10   if (std::abs (A[row][col]) < std::numeric_limits<double>::epsilon())
11     {
12       A[row][col] = std::accumulate
13         (A[row].begin (),
14          A[row].end (),
15          0.0,
16          [] (double sum,
17              const std::map<int, double>::value_type & p)
18          {
19            return (sum + std::abs (p.second));
20          }
21          );
22     }
23
24   if (! only_rhs)
25     A[row][col] *= 1e16;
26
27   // Multiply rhs by the diagonal entry
28   rhs[row] = A[row][col] * value;
29 }
```

Finally, the right-hand-side vector *sol* is assembled at each time step since it depends on the values of ρ at the previous time step.

```
1  for (auto quadrant = tmsh.begin_quadrant_sweep (); quadrant != tmsh.
    end_quadrant_sweep (); ++quadrant)
2  {
3      for (int ii = 0; ii < 8; ++ii)
4          if (! quadrant->is_hanging (ii))
5              g1[quadrant->gt (ii)] = sold[ord1(quadrant->gt (ii))];
6
7          else
8              for (int jj = 0; jj < quadrant->num_parents (ii); ++jj)
9                  g1[quadrant->gparent (jj, ii)] += 0.;
10 }
11
12 //rhs
```

```
bim3a_rhs (tmsh, f0, g0, sol, ord0);
bim3a_rhs (tmsh, f1, g1, sol, ord1);
```

2.3. Linear solver

The library we use to solve our linear system is MUMPS [1] ("MUltifrontal Massively Parallel Solver"), which is a package for solving systems of linear equations of the form $\boldsymbol{A}x = b$, where $\boldsymbol{A}$ is a square sparse matrix that can be either unsymmetric, symmetric positive definite, or general symmetric, on distributed memory computers. MUMPS implements a direct method based on a multifrontal approach which performs a Gaussian factorization $\boldsymbol{A} = \boldsymbol{LU}$, where $\boldsymbol{L}$ is a lower triangular matrix and $\boldsymbol{U}$ an upper triangular matrix. If the matrix is symmetric then the factorization $\boldsymbol{A} = \boldsymbol{LDL}^T$, where $\boldsymbol{D}$ is block diagonal matrix, is performed.
The system $\boldsymbol{A}x = b$ is solved in three main steps:

1. Analysis, during which preprocessing, including an ordering based on the symmetrized pattern $\boldsymbol{A} + \boldsymbol{A}^T$, and a symbolic factorization are performed.
2. Factorization, during which $\boldsymbol{A}_{\text{pre}} = \boldsymbol{LU}$ or $\boldsymbol{A}_{pre} = \boldsymbol{LDL}^T$, depending on the symmetry of the preprocessed matrix, is computed.
3. Solution, where the solution vector x_{pre} of $\boldsymbol{LU}x_{\text{pre}} = b_{pre}$ or $\boldsymbol{LDL}^T x_{\text{pre}} = b_{\text{pre}}$ is obtained through a forward elimination step $\boldsymbol{L}y = b_{\text{pre}}$ or $\boldsymbol{LD}y = b_{\text{pre}}$, followed by a backward elimination step $\boldsymbol{U}x_{\text{pre}} = y$ or $\boldsymbol{L}^T x_{\text{pre}} = y$. x_{pre} and b_{pre} are respectively the transformed solution x and right-hand side b associated to the preprocessed matrix $\boldsymbol{A}_{\text{pre}}$,

```
// Communicate matrix and RHS
A.assemble ();
sol.assemble ();

// Solver analysis
lin_solver->set_lhs_distributed ();
A.aij (xa, ir, jc, lin_solver->get_index_base ());
lin_solver->set_distributed_lhs_structure (A.rows (), ir, jc);
std::cout << "lin_solver->analyze () return value = "<< lin_solver->
   analyze () << std::endl;

// Matrix update
A.aij_update (xa, ir, jc, lin_solver->get_index_base ());
    lin_solver->set_distributed_lhs_data (xa);

// Factorization
std::cout << "lin_solver->factorize () = " << lin_solver->factorize ()
   << std::endl;

// Set RHS data
lin_solver->set_rhs_distributed (sol);

// Solution
std::cout << "lin_solver->solve () = " << lin_solver->solve () << std::
```

```
    endl;
```

2.4. Using the library

We report here the instruction to use the library, see the file *README.md* for more details.
In the main file *HVDC_main.cpp*, choose one of the tests by including the appropriate header file at lines 32-36. Then, compile the code with

```
make all
```

To generate the Doxygen documentation, run the command

```
make doc
```

Only the main file *HVDC_main.cpp* and the header file *Test_3.h* are included, the other header files have a similar structure so they are omitted.
The post processing is performed with two Octave functions included in the *script/m* folder. The first one, *export_phi_rho.m*, requires the function *export_tmesh_data.m* provided with bim++, so it is necessary to add the path *script/m* of bim++ to the Octave path. It generates the .vtu files, which can by opened using Paraview. The usage is the following:

```
export_phi_rho(<final time step>, <number of processes>)
```

with *final time step* being the index of the last time step and *number of processes* equal to the used number of processors.
Moreover, the file *rho.m* contains a function that generates a plot of the value of the charge density rho over time at the points specified in the .h file. The usage is the following:

```
rho(<refinement levels>,<cube side length>=0.001)
```

where *refinement levels* is a vector with the refinement levels of at the nodes selected by the *rho_idx* variable, and *cube side length* is the cubic domain dimension.

3 | Numerical Results

The following tests will be performed with the level 0 model 1.26 and the following values for some of the parameters:

$$L = 0.001, \quad V_0 = 15kV, \ \sigma = 3.21e-14 \quad \varepsilon_0 = 8.8542e-12$$

and as boundary condition we take 1.24.

3.1. Test 1

In the first test we consider a uniform value for permittivity equal to $\varepsilon = \varepsilon_r \varepsilon_0$, with $\varepsilon_r = 2$. The mesh has a uniform refinement of level $N_{\text{ref}} = 4$ as shown in Figure 3.1. The other parameters are set to $T = 15, \ \Delta t = 0.1, \ \tau = 2.0$.

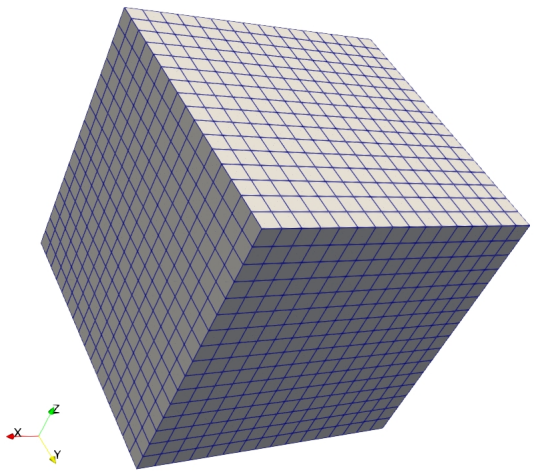

Figure 3.1: Test 1: uniform mesh, $N_{\text{ref}} = 4$.

The potential is uniform along the x and y axes, and it has a uniform gradient along the z axes (Figure 3.2).
We have an accumulation of positive charge on the top face and of negative charge on the bottom face, again uniform along the x and y axes, whereas in the remaining part of the domain the charge is zero (Figure 3.3).
The charge accumulation over time follows an asymptotic exponential behaviour with time constant equal to τ, as shown in Figure 3.4.

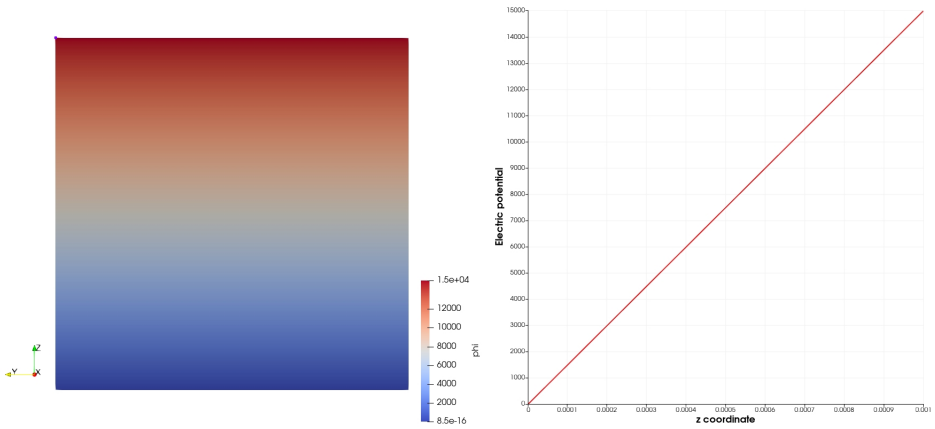

Figure 3.2: Test 1: electric potential in the cube (left) and as a function of the z coordinate (right) for $t = T$.

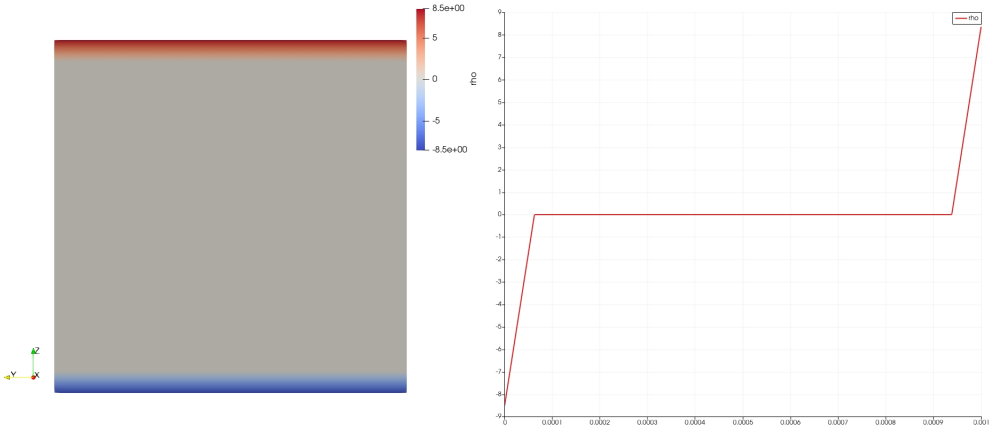

Figure 3.3: Test 1: charge density in the cube (left) and as a function of the z coordinate (right) for $t = T$.

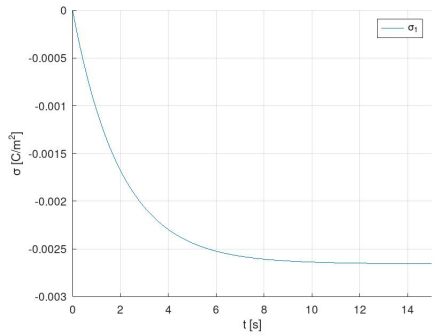

Figure 3.4: Test 1: charge density in the bottom face as a function of time.

3.2. Test 2

In the second test we consider two values for the permittivity: $\varepsilon_r = 2$ for $z < 0.0005$ and $\varepsilon_r = 4$ otherwise. The mesh has a uniform refinement of level $N_{\text{ref}} = 4$ plus a local refinement of level $N_{\text{ref,loc}} = 7$ along the plane $z = 0.0005$, see Figure 3.5.

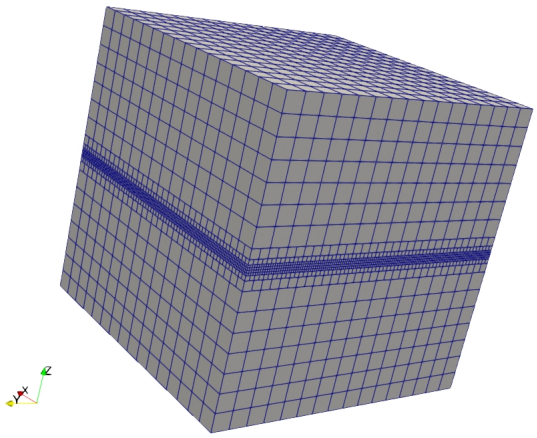

Figure 3.5: Test 2: uniform mesh, $N_{\text{ref}} = 4$, with local refinement $N_{\text{ref,loc}} = 6$ at $z = 0.0005$.

The resulting electric field is identical to the one obtained in Test 1, shown in Figure 3.2. At equilibrium, we have an accumulation of negative charge at the interface (Figure 3.6) as expected from theory [8].

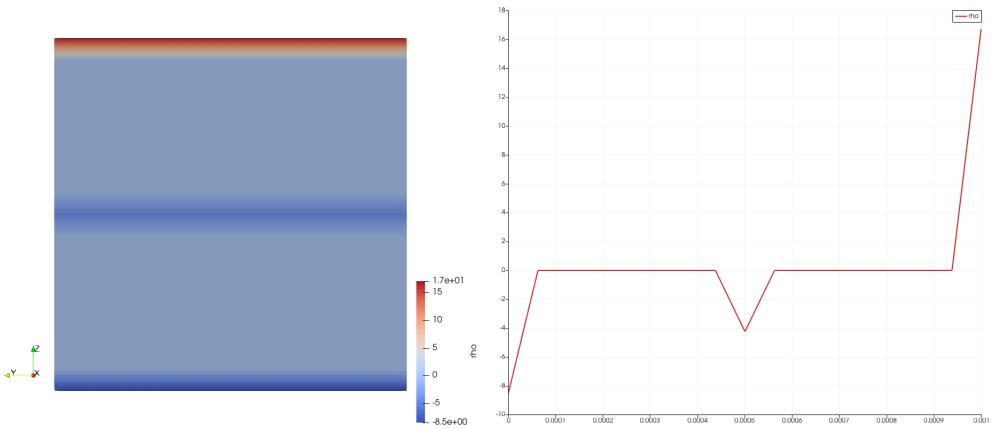

Figure 3.6: Test 2: charge density in the cube (left) and as a function of the z coordinate (right) for $t = T$.

Note the values of the plots of the type shown in Figure 3.6 depends on the local refinement level, meaning that the smaller is the grid element, the higher is the shown ρ value compared with a bigger element with the same charge density.
The time scale at which the charge accumulates varies significantly: we have first a fast

charge accumulation at the conductor interfaces, represented by σ_1 and σ_3 in Figure 3.7 left, followed by a slower movement of charge towards the interface at $z = 0.0005$ represented by σ_2 in Figure 3.7 right.

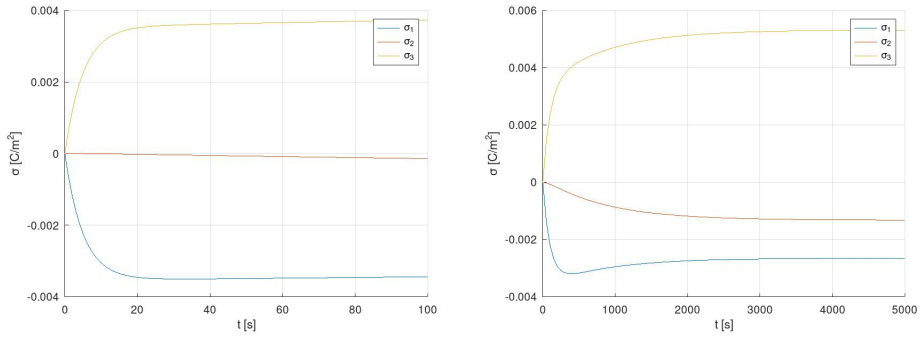

Figure 3.7: Test 2: charge density as a function of time: the left plot is computed with parameters $T = 100$, $\Delta t = 1$, $\tau = 5$, the right one with $T = 5000$, $\Delta t = 50$, $\tau = 50$. σ_1 represents the lower interface, σ_2 represents the central interface and σ_1 represents the upper interface.

3.3. Test 3

In Test 3 we try to simulate the behaviour of the insulator in mass impregnated cables, represented in Figure 3.8.

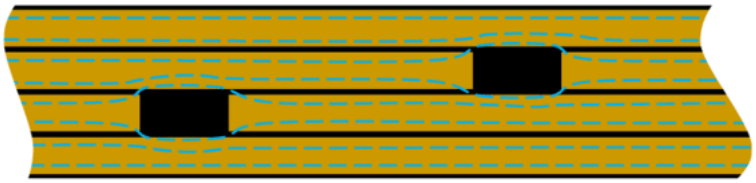

Figure 3.8: Test 3: structure of the insulation in a mass impregnated cable (credits: [6]).

We consider a cubic domain with three paper layers alternated by two oil layers, with the paper layer being 6 times thicker than the oil one. Moreover, we insert in the middle paper layer a cubic cavity, usually referred to as a butt-gap, filled with oil. The permittivity for the paper is $\varepsilon_r = 4$ and for the oil $\varepsilon_r = 2$. The mesh has refinement of level $N_{\text{ref.loc}} = 6$ at the interfaces between paper and oil, and $N_{\text{ref}} = 4$ elsewhere (Figure 3.9).
The resulting electric field is identical to the one obtained in Test 1, shown in Figure 3.2. The charge density is shown in figure Figure 3.10: we have an accumulation of charges at each interface, but not near the butt-gap since it oil filled and it is in contact with two oil layers. We don't have accumulation on the vertical faces of the butt-gap.
In Figure 3.11 we represent the charge values at different points over time. As before, the charge quickly accumulate on the top and bottom faces (σ_1 and σ_6), and then more slowly accumulates on the internal paper-oil interfaces.

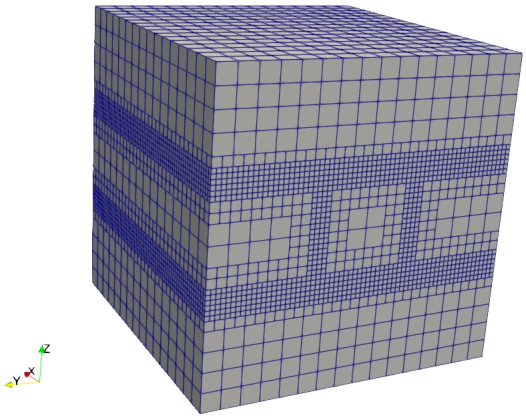

Figure 3.9: Test 3: mesh with local refinement of level $N_{\text{ref}} = 6$ at the paper-oil interfaces and at the border of the butt-gap, $N_{\text{ref}} = 4$ elsewhere.

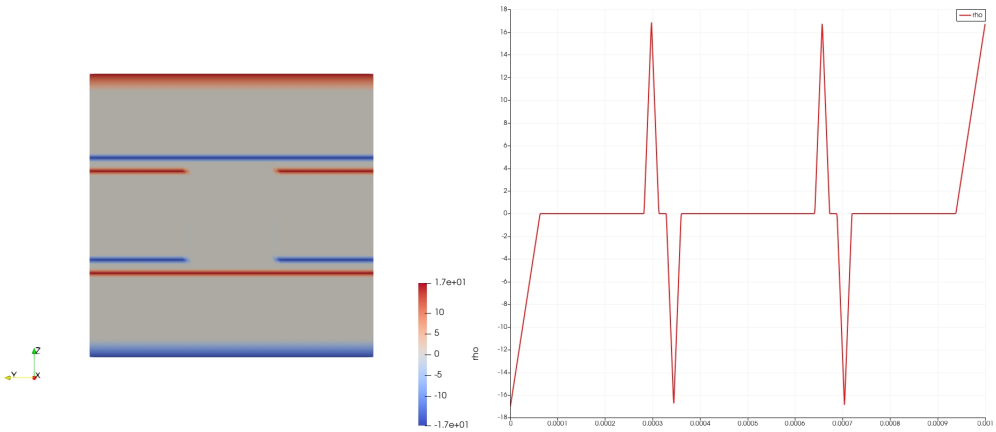

Figure 3.10: Test 3: charge density in the cube (left) and as a function of the z coordinate (right) for $t = T$.

3.4. Test 4

In Test 4 we consider an analogous scenario to Test 3, but this time the butt-gaps are filled with air, which we consider to have a permittivity value $\varepsilon_r = 1$ and conductivity $\sigma = 1e - 12$
The mesh is identical to the one used in Test 3. The electric field shows a sharp change in slope in the air filled butt-gap as shown in Figure 3.13.
The charge density is shown in figure Figure 3.14: we have an accumulation of charges at each interface, and also at the oil-air interfaces around the butt-gap. It is interesting to see that at these interfaces the sign of the charge is opposite compared to the paper-oil interfaces on the same z-plane. Moreover, contrary to the previous example, we have charge accumulation also on the vertical interfaces of the cavity. The charge density

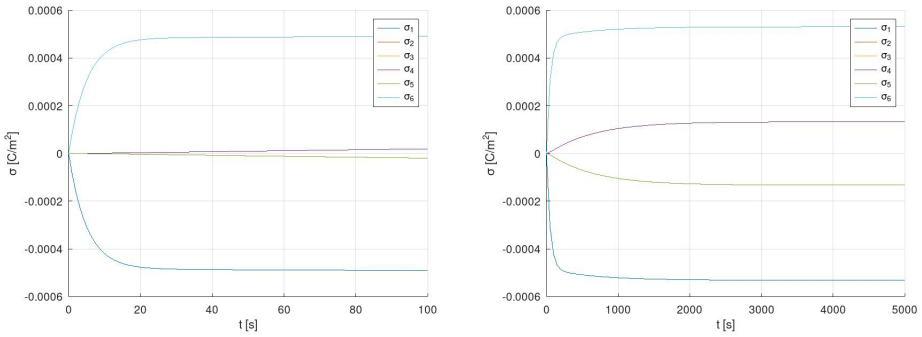

Figure 3.11: Test 3: charge density as a function of time: the left plot is computed with parameters $T = 100$, $\Delta t = 1$, $\tau = 5$, the right one with $T = 5000$, $\Delta t = 50$, $\tau = 50$. σ_1 represents the lower interface, σ_2 the first paper-oil interface, σ_3 the first oil-paper interface, σ_4 and σ_5 the second paper-oil and oil-paper interfaces and σ_6 represents the upper interface. Note that the plot for σ_2 overlaps with σ_4 and σ_3 with σ_5.

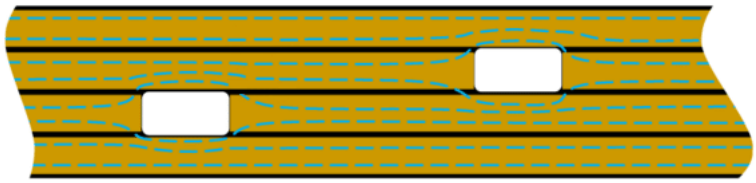

Figure 3.12: Test 4: structure of the insulation in a mass impregnated cable (credits: [6]).

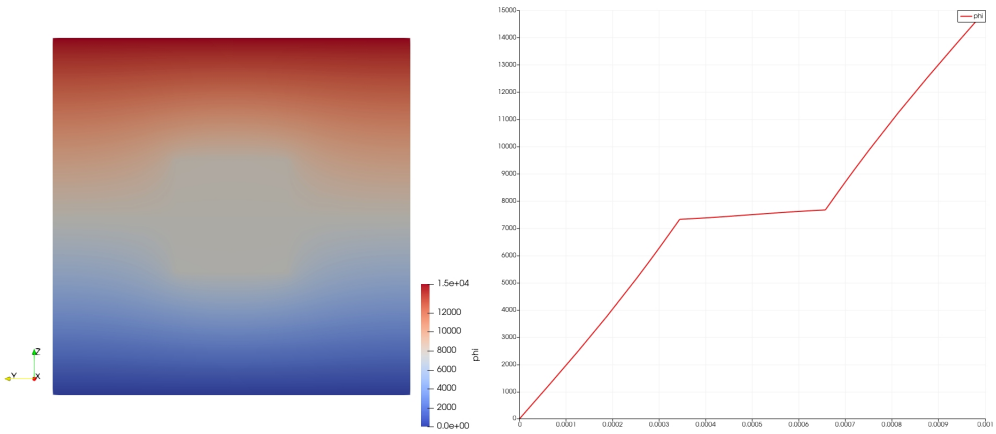

Figure 3.13: Test 4: electric potential in the cube (left) and as a function of the z coordinate (right) for a line that intersects the centre of the cavity and at $t = T$.

around the butt-gap is not uniform: if Figure 3.15 we show the values of ρ plotted over a line parallel to the x-axes, the first one passing through the bottom-left edge, the second one across the bottom face.
In Figure 3.16 we represent the ρ plotted over a line parallel to the y-axes, the first one passing through the front-bottom edge, the second one across the bottom face. From the

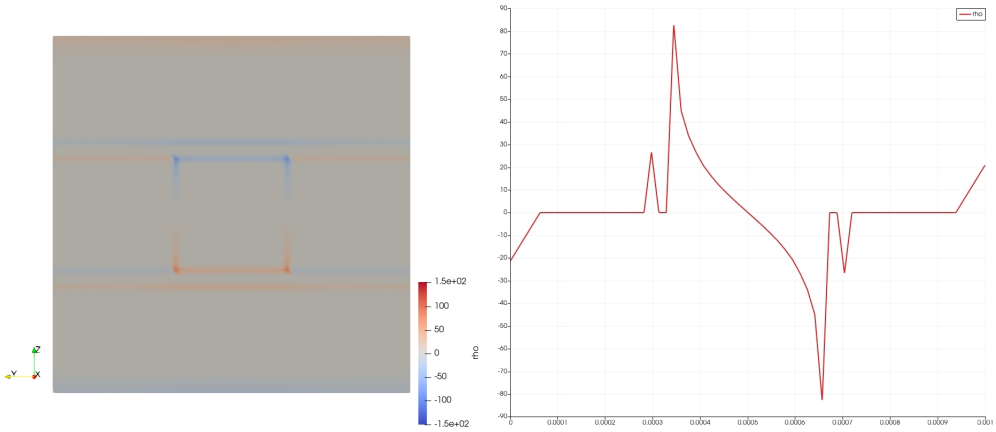

Figure 3.14: Test 4: charge density in the cube (left) and as a function of the z coordinate (right) for a line that intersects the left face of the cavity and at $t = T$.

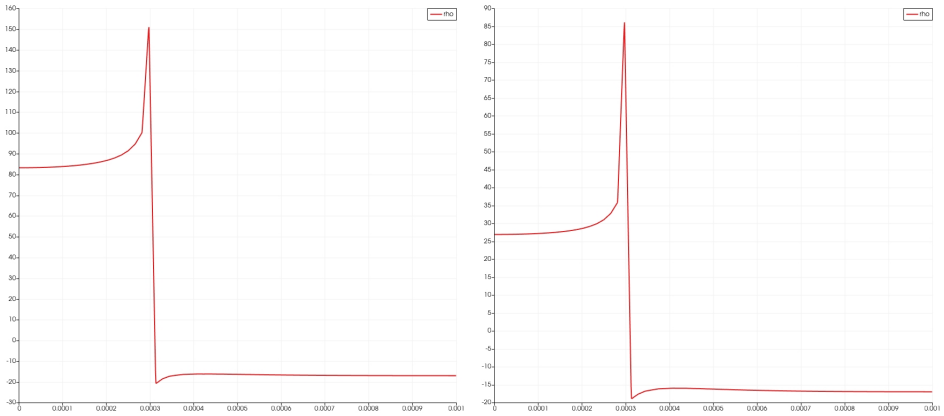

Figure 3.15: Test 4: charge density in the cube plotted over lines parallel to the x-axis at $t = T$.

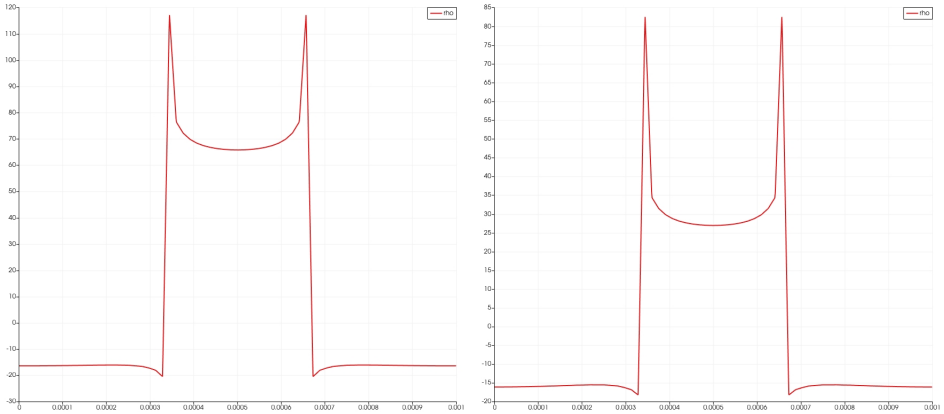

Figure 3.16: Test 4: charge density in the cube plotted over lines parallel to the y-axis at $t = T$.

previous plots we can see that we have a high charge accumulation at the vertices of the cavity and also at the air-oil interfaces.
In Figure 3.17 we represent the charge values at different points over time. We can see that the charge accumulate at different rates, with the top face (σ_1) being the fastest and the first paper-oil interface (σ_2) the slowest. Moreover, we can see that the highest ρ value is achieved at the back-bottom edge (σ_6), where oil, air and paper are present.

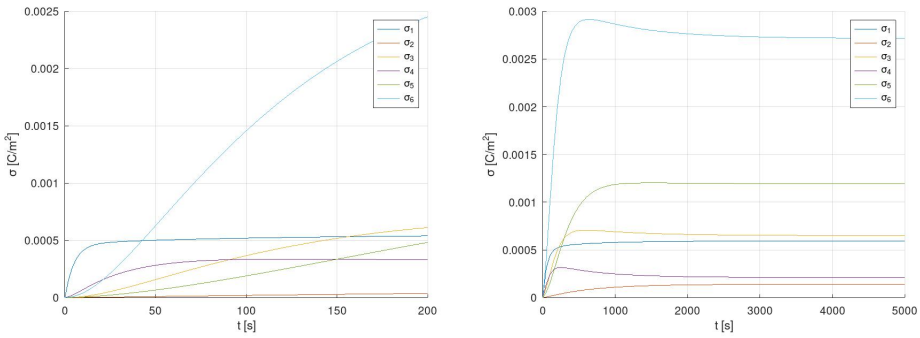

Figure 3.17: Test 4: charge density as a function of time: the left plot is computed with parameters $T = 200$, $\Delta t = 2$, $\tau = 5$, the right one with $T = 5000$, $\Delta t = 50$, $\tau = 50$. σ_1 represents the upper interface, σ_2 the first paper-oil interface, σ_3 the front-bottom-left vertex of the cavity, σ_4 the centre of the lower face, σ_5 the back-bottom-left vertex and σ_6 represents the centre of the back-bottom edge of the butt-gap.

3.5. Test 5

In Test 3 we to simulate the behaviour of o polymeric insulator with a spherical air-filled cavity inside, which represents the damage of the material due to the conduction current. We consider a cubic domain with homogeneous permittivity, but for a semi-spherical cavity filled with air, with parameters $\varepsilon_r = 1$ and $\sigma = 1e - 12$. The mesh has refinement of level $N_{\text{ref}} = 4$, and $N_{\text{ref,loc}} = 8$ around the sphere. (Figure 3.18).
The electric field is shown in Figure 3.19: we can see a sharp change of the gradient corresponding to the position of the sphere.
In Figure 3.20 we can see that we have an accumulation of charge on the border of the sphere. In particular, the charge density is greater in magnitude near the top and bottom of the sphere, and goes to 0 on the sides.
In Figure 3.21 we show that the charge accumulates more slowly on the sphere compared to the upper and lower interfaces of the domain.

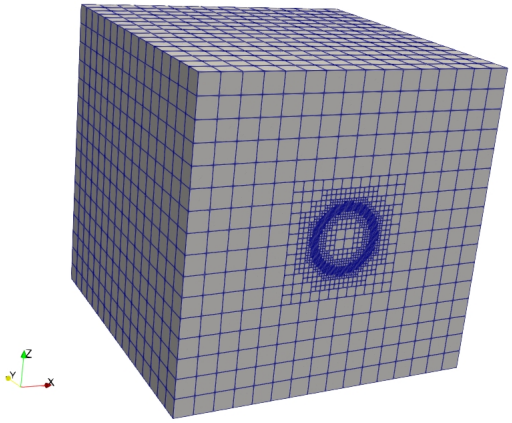

Figure 3.18: Test 5: mesh with local refinement of level $N_{\text{ref}} = 8$ around the sphere, $N_{\text{ref}} = 4$ elsewhere.

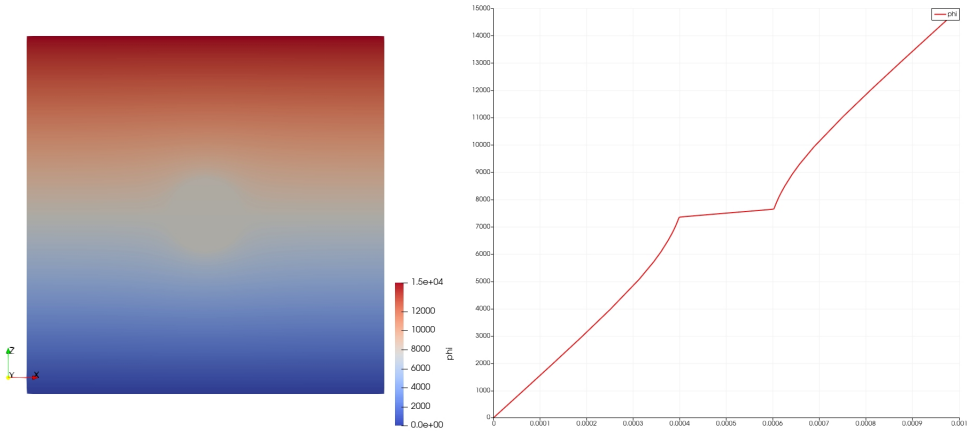

Figure 3.19: Test 5: electric potential in the cube (left) and as a function of the z coordinate (right) for $t = T$.

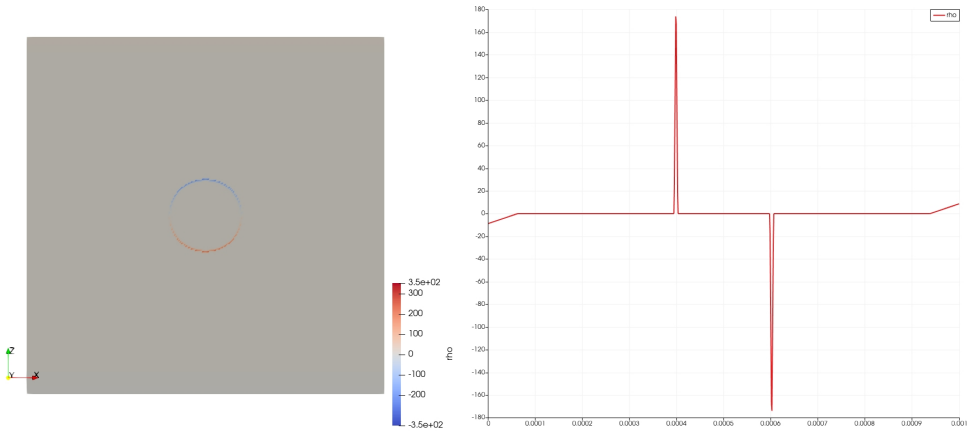

Figure 3.20: Test 5: charge density in the cube (left) and as a function of the z coordinate (right) for $t = T$.

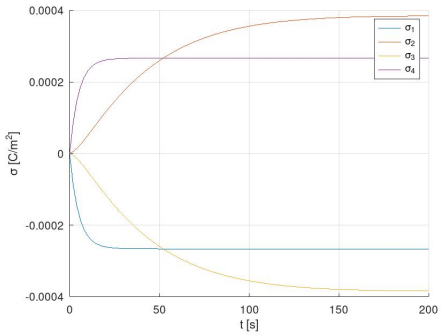

Figure 3.21: Test 5: charge density as a function of time: the plot is computed with parameters $T = 200$, $\Delta t = 2$, $\tau = 5$. σ_1 represents the lower interface, σ_2 the bottom and σ_3 the bottom and top of the sphere, σ_4 represents the upper interface.

4 | Conclusions and future developments

In this project we implemented a solver for a system of PDEs which models the charge and electric potential inside a dielectric material. The obtained results are consistent with the theoretical ones, showing the validity of the mathematical model and of the implementation. We have shown how the presence of air inside the material leads to a greater accumulation of charge locally compared to other interfaces such as paper-oil or two dielectric materials. The use of the bim++ library allows to speed up the computational time thanks to the parallelization, however more test on the scalability of the code are required.
The next step for this project would be to implement the higher complexity models Level 1 and Level 2, which are more computationally expensive but should yield result of higher accuracy when applied to polymeric insulators. Moreover, the analysis of the displacement and conduction currents could give a better understanding of the processes that lead to premature degradation of the materials involved.

Bibliography

[1] P. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[2] M. Ardelean and P. Minnebo. Hvdc submarine power cables in the world. *EUR 27527 EN*, 2015.

[3] P. Cambareri, C. de Falco, L. Di Rienzo, P. Seri, and G. C. Montanari. Simulation and modelling of transient electric fields in hvdc insulation systems based on polarization current measurements. *Energies*, 14(24), 2021. ISSN 1996-1073. doi: 10.3390/en14248323. URL `https://www.mdpi.com/1996-1073/14/24/8323`.

[4] D. Fabiani, G. C. Montanari, C. Laurent, G. Teyssedre, P. H. F. Morshuis, R. Bodega, L. A. Dissado, A. Campus, and U. H. Nilsson. Polymeric hvdc cable design and space charge accumulation. part 1: Insulation/semicon interface. *IEEE Electrical Insulation Magazine*, 23(6):11–19, 2007. doi: 10.1109/MEI.2007.4389975.

[5] A. K. Jonscher. Dielectric relaxation in solids. *Journal of Physics D: Applied Physics*, 32(14):R57–R70, jan 1999. doi: 10.1088/0022-3727/32/14/201. URL `https://doi.org/10.1088/0022-3727/32/14/201`.

[6] M. Runde, R. Hegerberg, N. Magnusson, E. Ildstad, and T. Ytrehus. Cavity formation in mass-impregnated hvdc subsea cables-mechanisms and critical parameters. *Electrical Insulation Magazine, IEEE*, 30:22–33, 03 2014. doi: 10.1109/MEI.2014.6749570.

[7] A. K. Upadhyay and C. C. Reddy. On the mechanism of charge transport in low density polyethylene. *Journal of Applied Physics*, 122(6):064105, 2017. doi: 10.1063/1.4997941.

[8] R. Zou, J. Hao, and R. Liao. Space/interface charge analysis of the multi-layer oil gap and oil impregnated pressboard under the electrical-thermal combined stress. *Energies*, 12(6), 2019. ISSN 1996-1073. doi: 10.3390/en12061099. URL `https://www.mdpi.com/1996-1073/12/6/1099`.