applied sciences

MDPI

Article

FLARE: A Framework for the Finite Element Simulation of Electromagnetic Interference on Buried Metallic Pipelines

Arturo Popoli *, Giacomo Pierotti, Fabio Ragazzi, Leonardo Sandrolini and Andrea Cristofolini

Department of Electrical, Electronic and Information Engineering, University of Bologna, 40136 Bologna, Italy; giacomo.pierotti2@unibo.it (G.P.); fabio.ragazzi4@unibo.it (F.R.); leonardo.sandrolini@unibo.it (L.S.); andrea.cristofolini@unibo.it (A.C.)

* Correspondence: arturo.popoli@unibo.it

Abstract: The functionality of buried metallic pipelines can be compromised by the electrical lines that share the same right-of-way. Given the considerable size of shared corridors, computer simulation is an important tool for performing risk assessment and mitigation design. In this work, we introduce an open-source computational framework for the analysis of electromagnetic interference on large earth-return structures. The developed framework is based on FLARE—an efficient finite element solver developed by the authors in MATLAB®. FLARE includes solvers for problems involving static electric and magnetic fields, and DC and time-harmonic AC currents. Quasi-magnetostatic transient problems can be studied through time-marching or—for linear problems—with an efficient inverse-Laplace approach. In this work, we succinctly describe the optimization of time-critical operations in FLARE, as well as the implementation of a transient solver with automatic time-stepping. We validate the numerical results obtained with FLARE via a comparison with the commercial software COMSOL Multiphysics® . We then use the validated time-marching analysis results to test the accuracy and efficiency of three numerical inverse-Laplace algorithms. The test problem considered is the assessment of the inductive coupling between a 500 kV transmission line and a metallic pipeline buried in the soil.

Keywords: numerical simulation; pipeline integrity; AC interference; earth-return; finite element analysis; inverse-Laplace; optimization

Citation: Popoli, A.; Pierotti, G.; Ragazzi, F.; Sandrolini, L.; Cristofolini, A. FLARE: A Framework for the Finite Element Simulation of Electromagnetic Interference on Buried Metallic Pipelines. *Appl. Sci.* **2023**, *13*, 6268. https://doi.org/10.3390/app13106268

Academic Editor: Andreas Sumper

Received: 7 April 2023
Revised: 15 May 2023
Accepted: 16 May 2023
Published: 20 May 2023

1. Introduction

Concerns over Metallic Pipelines Integrity

It is well known that high-voltage alternate current (HVAC) transmission lines can be the cause of significant electromagnetic interference effects on nearby metallic structures [1,2]. Coated pipelines for gas or liquid transportation are notable examples of this kind of structure. The resulting currents and voltages can lead to several kinds of unwanted effect. Workers coming into contact with the pipeline are subjected to shock hazards [3]; the connected cathodic protection systems and the pipeline itself can be damaged through accelerated aging and corrosion [4,5], as well as through perforations of its coating layer [6,7]. Pipelines can also be harmed by other sources of interference; for example, in recent years a number of works have been devoted to the effects produced by high-voltage direct current (HVDC) systems [8,9], as well as traction systems [10] and geomagnetic events [11]. Electrical stress on metallic pipelines may be caused by three coupling mechanisms, i.e., capacitive, inductive, and conductive coupling [12]. In this work, we focus on the most common coupling mechanism, i.e., inductive coupling. Overall, researchers have studied this topic for several decades, and many modeling efforts have been made throughout the years. Much of the initial efforts in this field originated from circuital or transmission line approaches based on Carson's formulae for mutual impedance [13–16]. These methods are also generally based on performing a

discretization of the considered corridor into sets of coupled smaller sections, described by lumped parameters [17]; such circuital approaches are computationally efficient, making them suitable for fast evaluations or parametric studies. In addition, the formulas for the approximate evaluation of mutual impedance have been extended over time to include the physical effects of soils constituted by multiple layers with different electrical resistivity values [18]. Nevertheless, over the years—thanks to the ever increasing availability of computing power—researchers have started to study the problem of pipeline integrity with field-theory methodologies [19,20]; these techniques are based on directly obtaining an approximated solution to Maxwell's equations for the electromagnetic field and allow accurate analysis of complex geometries and arbitrary material properties. In this way, gradually, models using the boundary element method (BEM) [21] or the finite element method (FEM) [22–24]—often in conjunction with circuit theory—have also been developed. An account of both recent and historic numerical contributions to inductive coupling modeling can be found in [25].

Several commercial codes dedicated to electromagnetic interference calculations also exist, featuring comprehensive modeling capabilities (also regarding problems related to grounding systems). These are often based on a combination of circuit theory and field methodologies, and allow the user to perform a variety of different analyses. Examples of such code programs include the well-established Elsyca [26], CDEGS [27], and XGSLAB [28].

The existing research literature on FEM models for AC interference calculations includes several works dedicated to harmonic analysis [20,22,29]. In this work, we describe an initial effort towards the development of a comprehensive FEM-based framework that can also deal with transient interference problems, which—to the best of the authors' knowledge—are currently more commonly assessed with transmission-line based approaches [30]. In addition, we also note that the validated research codes are generally closed-source and not available to other scientists and users in general.

For this reason, we introduce FLARE—Finite eLement Analysis foR Electromagnetics. FLARE is an open-source finite element code developed by the authors (FLARE is accessible at https://github.com/apopoli/FLARE (accessed on 30 March 2023)) that can be used to perform computational studies in several different areas of electromagnetics, including pipeline integrity assessment. This includes the assessment of electrical disturbances due to the inductive coupling between HVAC power lines and buried metallic pipelines. FLARE provides several solvers for problems involving static electric and magnetic fields, and DC and time-harmonic AC currents. The main novelties of this work are the vectorized implementation of two solvers for quasi-magnetostatic transient problems, the subsequent comparison and numerical validation of the two approaches, and the benchmarking of three inverse-Laplace routines included in FLARE. The approach based on the inverse-Laplace technique was introduced by the authors in [31]. A similar technique was also recently employed to study quasi-electrostatic problems in [32] and has been applied to other problems in the past, such as unsteady heat flow calculations [33]. The work in [31] was focused on the theoretical development of the method, and the developed code was based on a serial Fortran 90 inversion routine by D'Amore et al. [34]. In this work, we integrate the approach into the FLARE framework, providing a fully parallelized (shared memory) and more flexible MATLAB® implementation of the routines. We also add two additional (parallel) inversion routines, based on the work of Abate et al. [35]; as will be shown in a dedicated section of this work, these techniques are somewhat complementary to that by D'Amore and colleagues, from an accuracy and computational efficiency perspective. With respect to the work in [31], here we want to provide a numerical validation of the inverse-Laplace solution for a transient problem. To do that, we also implement a quasi-magnetostatic solver based on a time-marching procedure, which can operate with either fixed or variable time step lengths; we directly validate the time-marching solution against the solution yielded by the AC/DC module of COMSOL Multiphysics® on the

same problem. COMSOL Multiphysics® is also used to validate the complex domain time-harmonic solver in FLARE.

A concise description of the main features of the code is provided in Section 2, with emphasis on the vectorization process of the most time-critical operations. Then, in Section 3, we define a typical corridor geometry involving an HVAC line and a buried pipeline; we study the configuration with the time-harmonic solver in FLARE, comparing the solution to the one yielded by the AC/DC module of COMSOL Multiphysics®; we then use the quasi-magnetostatic time-marching solver to simulate a lightning event involving one of the phase conductors of a power line and repeat the numerical validation process using COMSOL Multiphysics®. We then show that the temporal evolution of the currents can also be obtained via the proposed inverse-Laplace routine, with the same level of accuracy. We finally perform a comparison of the accuracy and computational efficiency for the three numerical inversion routines implemented in FLARE.

2. Numerical Model

FLARE is a finite element code developed and implemented by the authors in MATLAB® [36]. Currently, FLARE includes several solvers for static and quasi-static problems over 2D planar domains. The domain discretization is performed by means of linear triangular elements. Future updates will include the option to address axysimmetric domains. Here follows a list of the available physical formulations.

2.1. Formulations

2.1.1. Static

Static electric and magnetic problems take the form of a Poisson equation:

$$\nabla \cdot (k \nabla \phi) = \chi. \tag{1}$$

For an electrostatic problem $\phi = \varphi$, $k = \epsilon_r$, and $\chi = -\frac{\rho}{\epsilon_0}$; φ is the scalar electric potential, ϵ_0 and ϵ_r are the dielectric constant of vacuum and the relative dielectric constant; ρ is the electric charge density. For a DC conduction problem—where the conduction current density vector field $\vec{J}$ is solenoidal—one has again $\phi = \varphi$, and $k = -\sigma$, obtaining $-\nabla \cdot (\sigma \nabla \varphi) = 0$, where σ is the electrical conductivity of the given medium. In this case one obtains the x and y components of the current density from $\vec{J} = -\sigma \nabla \varphi$.

The exact same formulation is retained for planar magnetostatic problems, where the current density and the magnetic vector potential A (MVP) have a preferential direction, e.g., z. In this case, referring to (1), $\phi = -A_z$, $k = 1/\mu_r$ and $\chi = \mu_0 J_z$, where A_z is the z (out of the page) component of the MVP field $\vec{A}$; μ_0 and μ_r are the magnetic permeability of vacuum and the relative magnetic permeability of the given medium, respectively; J_z is the z component of the current density.

2.1.2. Sinusoidal Steady-State (Time Harmonic)

For time-harmonic problems, the reported quasi-stationary unsteady formulation is simply transformed into a stationary complex problem in the frequency domain. This is performed by transforming the variables of the problem (whose time dependence is described by sinusoidal functions) into phasors, so that—for the given angular velocity ω—$A_z(x, y, t) \rightarrow \mathbf{A_z}(x, y)$; $\frac{\partial}{\partial t} \rightarrow j\omega$; We use bold notation to indicate phasors from here onward. In this way, the MVP formulation is written as

$$-\nabla \cdot \left(\frac{1}{\mu_r} \nabla \mathbf{A_z} \right) + j\omega\sigma\mu_0 \mathbf{A_z} = \mu_0(-\sigma \nabla \boldsymbol{\varphi}), \tag{2}$$

Given the source current density phasor $-\sigma \nabla \boldsymbol{\varphi} = \mathbf{J_{0,z}}$, the problem is reduced to finding the real and imaginary part of $\mathbf{A_z}$, from which all the other physical quantities of interest can be derived. The finite element discretization of the former equation is provided in [37].

2.1.3. Time-Domain

Quasi-stationary time-domain magnetic problems are solved in FLARE with a MVP formulation under the quasi-magnetostatic approximation ($\partial\vec{D}/\partial t = 0$, $\partial\vec{B}/\partial t \neq 0$):

$$-\nabla \cdot \left(\frac{1}{\mu_r}\nabla A_z\right) + \sigma\mu_0\frac{\partial A_z}{\partial t} = \mu_0(-\sigma\nabla\varphi). \tag{3}$$

Note that the previous expression—which is the time-domain version of (2)—reduces to (1) when $\partial/\partial t = 0$. The discretized problem is

$$[K][A_z] + [S]\left[\frac{dA_z}{dt}\right] = [T], \tag{4}$$

where $[A_z]$, $\left[\frac{dA_z}{dt}\right]$ and $[T]$ are arrays containing the nodal values of the MVP, its derivative, and the right-hand side of (3), respectively. $[K]$ and $[S]$ are assembled from the respective local matrices. Details on the employed vectorized assembly routine are provided in Section 2.2.

As anticipated, two different strategies can be adopted by the user to obtain the time-domain solution.

Time-Stepping Method

The first strategy is to use a time-marching strategy, based on the well-known Crank–Nicolson implicit method. The idea is to discretize the time-derivative in (4) by means of a centered finite difference scheme, as opposed to the forward or backward finite difference that would be employed for an explicit or implicit Euler scheme, respectively. In this way, one has for the given node i:

$$\frac{d}{dt}\left(A_{z,i}^{(k+1/2)}\right) \approx \frac{A_{z,i}^{(k+1)} - A_{z,i}^{(k)}}{\Delta t}, \tag{5}$$

where

$$A_{z,i}^{(k+1/2)} \approx \frac{1}{2}\left(A_{z,i}^{(k+1)} + A_{z,i}^{(k)}\right) \tag{6}$$

Substituting (5) and (6) in (4), one finds:

$$\frac{1}{2}[K]\left([A_z]^{(k+1)} + [A_z]^{(k)}\right) + \frac{1}{\Delta t}[S]\left([A_z]^{(k+1)} - [A_z]^{(k)}\right) = [T]^{(k+1/2)}. \tag{7}$$

Then, factoring out the terms at time instants $(k+1)$ and (k) one obtains the final expression:

$$[M_1][A_z]^{(k+1)} = [T]^{(k+1/2)} + [M_2][A_z]^{(k)}, \tag{8}$$

where

$$[M_1] = \left[\frac{1}{2}[K] + \frac{1}{\Delta t}[S]\right]; \quad [M_2] = \left[\frac{1}{\Delta t}[S] - \frac{1}{2}[K]\right] \tag{9}$$

The Crank–Nicolson scheme is unconditionally stable and second-order accurate in time [38].

The code can be run using either constant or variable time step lengths. When a constant time step is selected the code uses the `decompose` MATLAB® function on the matrix $[M_1]$ in (8). `decompose` performs an automatic selection between several matrix decomposition algorithms, based on the properties of the provided matrix. We refer the reader to the MATLAB® guide for further details on the adopted procedure [39]. While the decomposition procedure can be time-intensive for matrices with large numbers of non-zero entries, we found it to be quite efficient for the sizes considered in this work. For example, `decompose` took ≈0.41 s for the sparse matrix that will be considered in Section 3, having rank $\approx 7 \times 10^4$ and $\approx 5 \times 10^5$ non-zero entries. In any case, the decomposition of the

matrix is performed only once per simulation, and used at each step of the time-loop to compute $[A_z]^{(k+1)}$ without repeating the decomposition phase of the (direct) linear system solving procedure. This, at least for the use-cases explored by the authors in this work, allows for an execution time reduction of approximately one order of magnitude when using `decompose`. Conversely, when a variable time step is selected, the Δt in the left-hand side of the time-discrete version of (8) changes at each iteration. Hence, `decompose` cannot be used in this case. Nevertheless, using an adaptive time step can be extremely advantageous for exponentially varying sources. The time step used to obtain the solution at the next time instant $(k+1)$ is obtained from the one chosen at the previous instant (k), based on the maximum local relative variation of the MVP:

$$\Delta t^{(k+1)} = \frac{\Delta t^{(k)}}{\delta/\delta_{\text{rel}} + \delta_{\text{abs}}}, \tag{10}$$

where δ_{rel} and δ_{abs} are selected by the user to set the sensitivity of the adaptivity to temporal variations of the solution. The variable δ provides an indication of the maximum relative change of the solution in the computational domain:

$$\delta = \max\left[\frac{|[A_z]^{(k)} - [A_z]^{(k-1)}|}{0.5|[A_z]^{(k)} + [A_z]^{(k-1)}| + \varepsilon}\right]. \tag{11}$$

The constant quantity $\varepsilon \ll \delta_{\text{abs}}$ is an arbitrary small quantity needed to avoid a zero-division in the above expression.

Inverse-Laplace Transform Method

As anticipated, FLARE integrates a MATLAB translation of the Fortran 90 routine for the numerical inverse-Laplace transform developed by D'Amore and colleagues [34,40]. This allows one to compute the time-evolution of one of the quantities yielded by the FEM solver, using solutions computed in the complex domain. This formulation is closely related to the quasi-stationary formulation, substituting the purely imaginary $j\omega$ with the complex variable s. Details of this approach can be found in [31] and are briefly summarized below.

The Laplace transform $\mathcal{L}\{f(t)\}$ of a generic function of time $f(t), t \geq 0$ is [41]:

$$F(s) = \int_0^\infty e^{-st} f(t)\, dt, \tag{12}$$

The analytical inversion of the Laplace transform can be obtained via the Bromwich (or Riemann) inversion theorem:

$$f(t) = \frac{1}{2\pi i} \int_{\gamma - i\infty}^{\gamma + i\infty} e^{st} F(s)\, ds, \tag{13}$$

where γ—the real part of s—is such that the contour of integration is to the right of any singularities of $F(s)$ greater or equal to the abscissa of convergence σ_0.

Unfortunately, the analytical evaluation of (13) is only straightforward for a limited number of cases. For this reason, great research efforts have been dedicated to the development of numerical inversion algorithms. Notably, at the present stage, it is not possible to define the best strategy for all possible cases, and the performance of different inversion algorithms depends on the particular application [42]. Due to this, we provide three different inversion techniques in FLARE. These are compared for the case considered in this work in Section 3.6.

2.2. Vectorized Right-Hand Side Assembly Routine

Transient solvers based on time-stepping algorithms require a more careful implementation with respect to stationary solvers, since many operations must be performed multiple times; typically, once every time step. The postprocessing of the computed variables may

also be critical for both the execution time and required memory, especially when dealing with meshes with many nodes or time steps. Aside from the solution of the sparse linear system arising from the finite element discretization of the physical formulation, one of the most time-consuming operations of a FEM solver is the assembly of the global matrix and the right-hand side from the element matrices. Modern interpreted languages such as Python or MATLAB® usually provide wrappers for optimized C, C++, or FORTRAN routines for the solution of linear systems; naive assembly techniques are instead based on `for` loops, which notoriously exhibit low computational efficiencies in interpreted languages [43]. For this reason, assembly operations often constitute the main performance bottleneck in MATLAB® finite element codes.

In order to make the assembly procedure efficient, the idea introduced in [44] is exploited in FLARE. The terms of all the local element matrices (i.e., the local contributions of each element to the global solution) are computed with a vectorized procedure instead of a `for`-based procedure. Since the local element matrices are 3×3 matrices, the contributions must be stored into a $9 \times n_{el}$ array, where n_{el} is the total number of elements of the given mesh. Two auxiliary arrays with the same size are also used to provide a mapping between the local and global views. The arrays store the row and column indices of the global matrix corresponding to each element of the local values. In this way, the entries of the local contribution array are assembled into the global matrix with a single call to the `sparse` MATLAB® Matlab function. A comprehensive performance comparison of vectorized assembly procedures in different languages can be found in the work of Cuvelier and colleagues [45].

By preassembling the two matrices $[K]$ and $[S]$ in (9), the square sparse matrix $[M_1]$ in (8) can be obtained at each time step by summing $[K]$ and $[S]$ after calculating the element-wise products by $1/2$ and $1/\Delta t$, respectively. The same technique can be adopted for $[M_2]$ multiplying the solution array at the previous time instant. Unfortunately, the array $[T]^{(k+1/2)}$ must be reassembled at each time step with a time-varying $J_{0,z}$ source term.

In order to avoid this operation, we define an auxiliary matrix $[K_t]$ with size $n_p \times n_{el}$, where n_p is the number of points (vertices) of the mesh. The aim of $[K_t]$ is to allow the array $[T]^{(k+1/2)}$ to be computed as a matrix–vector product after computing the source term at the considered time instant. The array $[Q]$ of size $n_{el} \times 1$ contains the quantity that each element contributes to each of the three nodes representing its vertices. $[Q]$ is defined by the (weighted) element-wise product between the column-arrays $\left[J_{0,z}^{(k+1/2)}\right]$ and $[Aa]$:

$$[Q] = \frac{1}{3}\mu_0 \left[J_{0,z}^{(k+1/2)}\right] \odot [Aa], \tag{14}$$

where $\left[J_{0,z}^{(k+1/2)}\right]$ and $[Aa]$ store the element current densities evaluated at the time instant $(k+1/2)$ and the element areas, respectively. Let us consider the simple two-element and four-node mesh in Figure 1. Given the represented element and node numbering, the connectivity matrix can be written as

$$[C] = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 4 \end{bmatrix}. \tag{15}$$

The entries of the auxiliary matrix $[K_t]$ are either zeros or ones. The nonzero elements of the ith column of $[K_t]$ correspond to the ith row of the connectivity matrix $[C]$. In this way, for the considered example, one has

$$\left[T^{(k+1/2)}\right] = [K_t][Q] = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \frac{1}{3}\mu_0 \begin{bmatrix} J_{z0,1}^{(k+1/2)} Aa_1 \\ J_{z0,1}^{(k+1/2)} Aa_1 + J_{z0,2}^{(k+1/2)} Aa_2 \\ J_{z0,2}^{(k+1/2)} Aa_2 \\ J_{z0,1}^{(k+1/2)} Aa_1 + J_{z0,2}^{(k+1/2)} Aa_2 \end{bmatrix}. \quad (16)$$

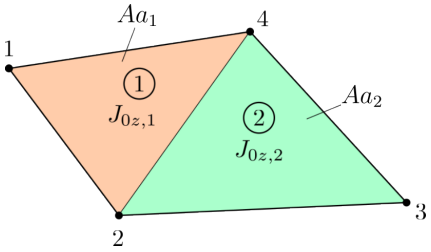

Figure 1. Example of a triangular mesh with two elements and four nodes.

The assembly procedure of $[K_t]$ is performed using the `sparse` function.

```
%% Assembly of K_t auxiliary matrix
% concatenated rows of the connectivity matrix C
ii = reshape(C',[], 1);
% n_el: number of mesh elements
jj = repelem(1:n_el,3); % 1 1 1 2 2 2 3 3 3 ...
% n_p: number of mesh points
K_t = sparse(ii,jj,1,n_p,n_el); % assemble ones in a n_p X
    n_el matrix with rows and column indices given by ii and jj
     arrays
```

2.3. Software Structure

This section provides a description of the structure and variable handling of FLARE. The code is designed for flexibility, while keeping a consistent organization of the settings, regardless of the solver chosen by the user. Let us consider one of the simplest possible problems, computing the electric potential on a uniformly charged unit disk. The basic structure of a simulation is provided in Figure 2. One starts by generating the computational mesh used to perform the calculation. This can be done by either loading a file that has been previously generated by `Gmsh` or by calling (from MATLAB®) the Python `Gmsh` APIs. In the first case, if the mesh from a `.geo` file called, e.g., `. unit_circle .geo` has been exported in `.m` format, the information can be loaded via a single command, e.g.,

```
unit_circle; % import mesh file
ndom = num_regions(msh); % count number of regions (domains)
```

Using the Python `Gmsh` APIs allows dynamically providing instructions to `Gmsh` from FLARE , and it is advised when the mesh must be updated during the simulations, such as when geometry optimizations are performed. In both cases, a `msh` MATLAB® structure file is generated, containing the information on the computational mesh. This is one of the two inputs to the `fesolve` function used to obtain the solution.

The next phase (settings) involves filling several fields of the structure `opts`, which will be passed to the solver. Here is an example of the considered electrostatic problem on the unit disk:

```
opts.tag_boundary = 1; % set domain boundary on edges marked
    with tag=1 in msh
[opts.materials] = set_materials('mesh_unit_circle',ndom); %
    define material properties
```

```
opts.ProblemKind = 'Electrostatic'; % [Electrostatic][
    Magnetostatic][QMagnetostaticSin][MagTimeDependent] % set
    the kind of problem to be solved
opts.source = 1; % unit charge density, uniformly distributed
```

The set_materials function requires a string containing the test case name—for example 'mesh_unit_circle'—and returns a string array named `materials`, containing the material for each region of the domain:

```
function [material] = set_materials(flag,ndom)
switch flag
case ('mesh_unit_circle')
material(1) = 'air';
...
```

The properties of the material `'air'` are stored in a separate function called `MatLib`:

```
function [Mprop] = MatLib(MatKind)
% prop(1) = material relative permittivity [Adim]
% prop(2) = material relative permeability [Adim]
% prop(3) = material electric conductivity [S/m]
switch MatKind
case "air"
Mprop = [1, 1, 0];
...
```

While the source term in the above code snippet has been set to 1, FLARE easily allows for more sophisticated source terms that can vary over time and depend on the given mesh region. One may indeed write a standalone MATLAB® function in a separate file and pass a function handle to FLARE. Figure 2 shows an example of this method for a source term representing a lightning current.

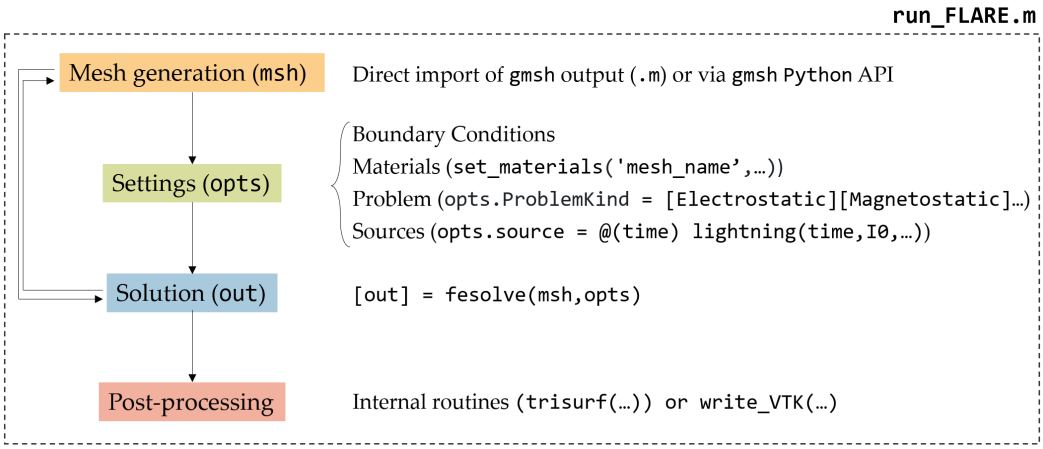

Figure 2. Structure of a typical FLARE simulation.

Regardless of the selected solver, one always calls the same function to obtain the solution, which is essentially a driver making a series of subcalls depending on the selected options:

```
[out] = fesolve(msh,opts); % setup and compute solution
```

When a stationary solver is called, `out` file structure has only two fields, named `field` and `scal`. These store information about the field variables, such as the MVP A_z or the current density J_z, or scalar variables, such as the current of each region (surface integral of J_z). When a transient solver is selected, the two former fields only store the information computed at the latest time instant, and `out` is populated with more fields. The field `tm`,

which stands for time measurement, stores the cumulative computation time spent during several key processes, such as the matrix assembly or the linear system solution. Variables saved over time are stored in a dedicated field called `sv`.

Finally, the user has two options regarding visualization. First, MATLAB® includes an extensive suite of plotting tools, including functions dedicated to unstructured meshes, such as the `trisurf` function:

```
x = msh.POS(:,1); y = msh.POS(:,2); % get mesh coordinates
trisurf(msh.TRIANGLES(:,1:3),x,y,out.field.phi,out.field.phi)
```

Nevertheless, the native MATLAB® routines might not be fully adequate for transient problems over large domains. For this case, we integrated a binary export tool in `.vtk` format, which can be read by the popular open-source visualization tool `ParaView`.

3. Results and Discussion

In the following sections, we consider a typical corridor configuration, where a metallic pipe is buried within the soil under an HVAC transmission line. In particular, we consider a single section of the corridor in a 2D planar domain. The choice of a 2D approach allows for a direct comparison between the solvers in FLARE and the ones implemented in COMSOL Multiphysics® [46], which we use as a benchmark for FLARE. However, this simplifying assumption corresponds to the assumption that each considered conductor has an ideal return path. We refer the reader to the more in-depth discussion of this limitation of 2D approaches provided in [37,47], and we save the implementation in FLARE of the numerical techniques described herein for future works.

We simulated the considered configuration, focusing on the current induced along the pipeline for two different operating conditions of the transmission line, i.e., sinusoidal and transient. These two physical scenarios required different numerical approaches implemented in FLARE (see Section 2). In each case, the results yielded by FLARE were validated against COMSOL Multiphysics®, a well-established commercial software that has already been used to analyze electromagnetic interference on pipelines in [48].

3.1. Corridor Geometry

The transmission line geometry used in this work consists of a single-circuit line and two overhead ground wires (OGWs). This configuration—typical of 500 kV AC systems used in the U.S.—was taken from the work of Yao and colleagues in [49]. The line and the metallic pipeline buried in the soil are shown in Figure 3. As one can see, the electrical line and the pipeline are embedded into two semicircular regions with radius 5 km. The MVP A_z is assumed to be null at the outer edge of the domain (Dirichlet boundary condition). The transmission line conductor's sizes, not specified in [49], were taken from [50], where a similar configuration was considered. The radii of the phase conductors and the OGWs are 15.3 mm and 5.6 mm. The pipeline has a diameter of 1 m and a thickness of 1.5 cm. Regarding the electric properties of the materials, the three phase conductors are made of aluminum ($\sigma_{\text{phase}} = 3.5 \times 10^7$ S/m), while the OGWs and the pipeline are made of steel ($\sigma_{\text{OGW,pipe}} = 5.5 \times 10^6$ S/m). The electrical conductivity of the soil is $\sigma_{\text{soil}} = 1 \times 10^{-2}$ S/m.

The domain in Figure 3 was discretized with the open-source `Gmsh 4.11.1` software [51], which allows one to easily embed structured regions in a non-structured Delaunay triangulation. This is particularly useful for capturing the current density distribution in highly conductive (or magnetically permeable) regions subjected to the skin effect, especially at frequencies typical of transient phenomena. The mesh has 70,350 nodes and 140,568 triangles. The characteristic size of the elements ranges from ≈250 m at the outer edge to $\approx 1 \times 10^{-6}$ m at the surface of metallic conductors (e.g., the buried pipeline). At the soil surface a coarser mesh size can be used, due to the soil electrical conductivity being several orders of magnitude smaller than typical values for metallic conductors; yielding a larger skin depth.

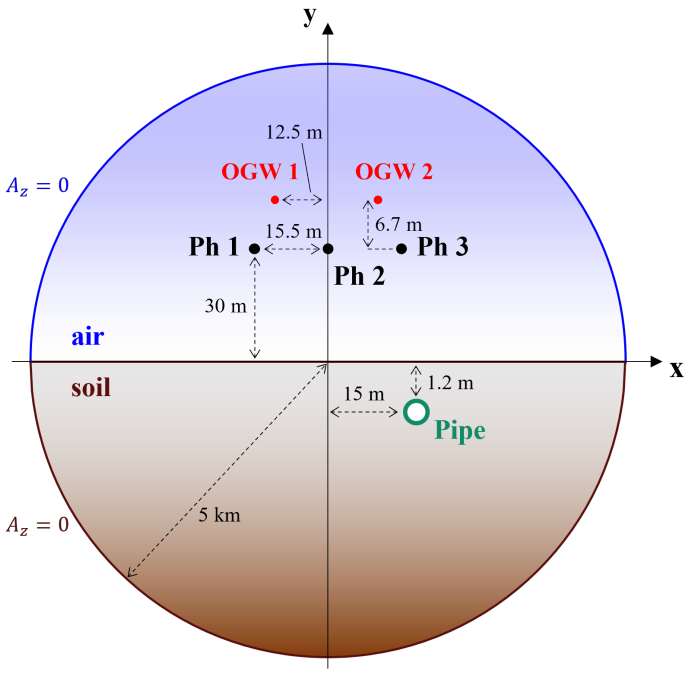

Figure 3. Geometry of the considered right-of-ways (not to scale).

3.2. Time-Harmonic Analysis

The geometry described in Section 3.1 was first studied by means of a time-harmonic analysis at 60 Hz, using the complex MVP described in Section 2.1.2. This allowed the induced interference levels in normal operating conditions of the power line to be examined. The described formulation allowed the value of J_0 to be enforced for each element of the domain. J_0 represents the current density that one would find in a given element for a stationary problem, i.e., neglecting the effects of Faraday's law. For a problem with frequency $f \neq 0$, the total current density is given by

$$J(t) = J_0(t) - \sigma \frac{\partial A_z(t)}{\partial t}, \tag{17}$$

which for the time-harmonic formulation becomes:

$$\mathbf{J} = \mathbf{J_0} - j\omega\sigma\mathbf{A_z}. \tag{18}$$

To simulate a single section of a 250 A balanced three-phase system using a current-driven approach, we set $\sigma = 0$ on the phase conductors and we enforce $\mathbf{J_0} = 250/S_{\text{phase}} e^{-ik2/3\pi}$, where $k = 1, \ldots, 3$ for phase conductors 1, 2, and 3, respectively. S_{phase} is the phase conductor cross-section. The same result can be obtained with a voltage-driven approach, using the real electrical conductivity values for the three phase conductors and by subsequently setting $J_{0,z}$ such that the total phase currents (determined by $J_{0,z}$ and $-j\omega\sigma A_z$) have the desired value.

Since FLARE is coded in MATLAB®, the obtained finite element solution for a given problem can be easily visualized using the `trisurf` function. Field lines can be also drawn using the `streamslice` utility. The magnitude of the computed MVP near the three-phase conductors and around the buried pipeline is plotted in Figure 4. The magnetic field lines are also plotted at $t = 1$ ms. A distortion of the field lines path can be noticed at the OGWs and pipeline location, caused by the local effects of the magnetic field produced by the induced currents. One can also see that, at this frequency and for the considered value of soil resistivity, the magnetic field screening effect exerted by the soil was low. This was expected, since the soil penetration depth $\delta = \sqrt{(\pi f \sigma \mu)^{-1}}$ was on the order of 10^2 m for the considered conditions.

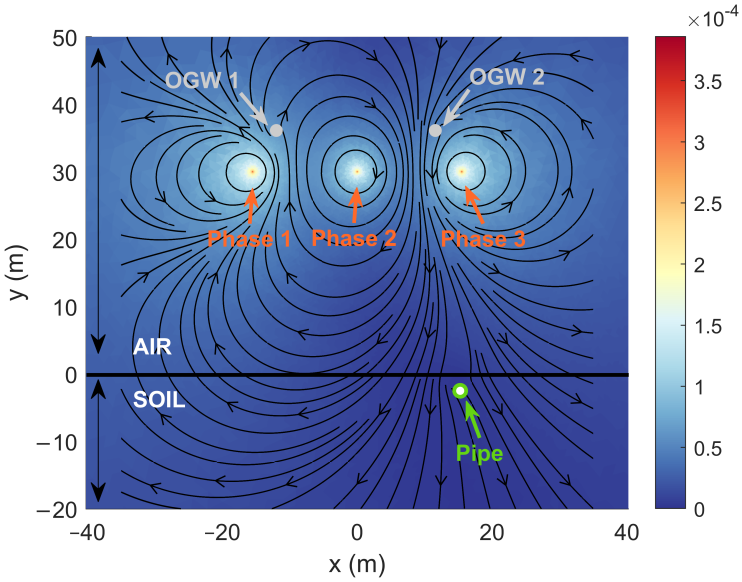

Figure 4. Magnitude of the MVP z- component $A_z(x, y)$ in $\mathrm{T\,m^{-1}}$ and magnetic field lines when $I_{\text{phase 1}} = 250\,\mathrm{A}$, $t = 1 \times 10^{-3}\,\mathrm{s}$. The OGWs and the pipeline are not to scale.

In order to validate the time-harmonic solver, we compared the obtained solution with that yielded by the well-established COMSOL Multiphysics® software. More details of the simulation in COMSOL Multiphysics® are provided in Section 3.4. The results are summarized in Table 1, where the magnitude and phase of the induced current are compared for the considered conductors. The column marked with $\Delta_r\%$ shows the relative percentage difference between the two codes for the given quantity. This quantity is defined as

$$\Delta_r\% = |(I_{\text{FLARE}} - I_{\text{COMSOL}})/I_{\text{COMSOL}} \cdot 100|. \tag{19}$$

For the current magnitude and phase, the largest differences were below 1%.

Table 1. Time-harmonic case (60 Hz)—comparison between the currents computed with FLARE and COMSOL Multiphysics®.

Region	Current Magnitude (A)			Phase (deg)		
	FLARE	COMSOL	$\Delta_r\%$ [1]	FLARE	COMSOL	$\Delta_r\%$ [1]
OGW 1	12.2731	12.2730	0.0004	−140.8317	−140.9354	0.0736
OGW 2	10.2212	10.2205	0.0074	42.3653	42.2765	0.2100
pipe	10.4912	10.4877	0.0332	−10.7012	−10.7105	0.0864
Soil	8.1174	8.1091	0.1023	166.3202	166.3265	0.0038

[1] Percentage difference, given by (19).

3.3. Lightning-Induced Phase Conductor Current

The aim of this section was to consider a transient problem of practical relevance, from the perspective of both transmission line modeling and pipeline integrity. Yao and colleagues used a finite-difference time domain methodology in [49] to simulate the effects of a lightning strike on a 500 kV transmission line tower. The authors computed the lightning-induced current on one of the phase conductors. We assumed that the same current flows through phase 1 of the transmission line described in Section 3.1, and we used two different numerical strategies to assess the transient response of the whole system

under these conditions. We used a Heidler function [52] to represent the phase 1 current from [49], i.e.,

$$i_{\text{phase 1}}(t) = -\frac{I_0}{\eta}\frac{(t/\tau_1)^n}{1+(t/\tau_1)^n}\exp\left(-\frac{t}{\tau_2}\right), \tag{20}$$

where $I_0 = 1 \times 10^4$ A, $\tau_1 = 1.25 \times 10^{-6}$ s, $\tau_2 = 5.5 \times 10^{-5}$ s, $\eta = 0.875$, and $n = 2$.

We started by simulating the transient response of the system to $i_{\text{phase 1}}(t)$ with the time-domain solver in FLARE and COMSOL Multiphysics® over 1 ms. Concerning FLARE, the automatic time-stepping procedure described in Section 2.1.3 led to 158 time steps. The time step lengths ranged from $\Delta t \approx 1 \times 10^{-13}$ s at the beginning of the simulation when $i_{\text{phase 1}}(t)$ increased rapidly to $\Delta t \approx 1 \times 10^{-4}$ s in the latter stages of the $i_{\text{phase 1}}(t)$ decay. The total execution time (wall-clock time) of the simulation, including the postprocessing, was $t_{\text{WC}} \approx 135$ s on the CPU used for testing (AMD Ryzen 7 PRO 4750U). In comparison, COMSOL Multiphysics® (which also uses an adaptive time-stepping technique) performed the simulation with approximately half the time steps, requiring ≈45 s.

The time evolution of the currents computed with FLARE is shown in Figure 5 over a limited portion of the simulation time. The gray continuous line represents $i_{\text{phase 1}}(t)$, enforced on phase 1. The square markers on $i_{\text{phase 1}}(t)$ mark the time instants used for the computation.

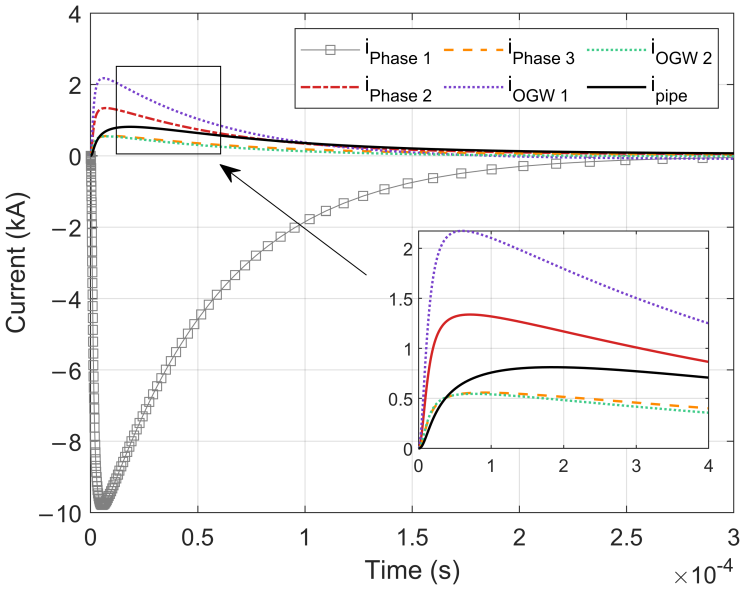

Figure 5. Transient behavior of the currents on a cross-section of the geometry of Figure 3, where $i_{\text{phase 1}}$ (gray continuous line) is given by (20). The square markers on $i_{\text{phase 1}}$ mark the time steps used by the automatic time-stepping algorithm.

All the obtained currents were positive, in contrast to $i_{\text{phase 1}}(t)$. While the pipeline had a lower electrical conductivity with respect to the phase conductors and was partially screened by the soil, its maximum induced current was larger than the one in phase 3. This was due to a combination of its location in close proximity to the transmission line and its surface, considerably larger than the one of either the phase or overhead ground conductors.

In order to verify the accuracy of the solution obtained with FLARE, we compared a subset of the currents shown in Figure 5 to the same quantities evaluated over time using the COMSOL Multiphysics® software. We also highlight that the FLARE wall-clock time was comparable or lower to that needed to perform the same simulation in COMSOL Multiphysics® on the same machine.

The values of $i_{\text{phase 2}}(t)$, $i_{\text{phase 3}}(t)$, and $i_{\text{pipe}}(t)$ yielded by the two approaches are shown in Figure 6. Differently from Figure 5, the currents have been plotted using a

logarithmic scale for the temporal axis, to better appreciate the agreement of the two approaches in the first part of the simulation, for small values of t. The markers indicating the FLARE solutions are only shown for one value out of three, for the sake of clarity. The agreement was fully satisfactory over the whole simulated time span. Focusing on the maximum value of $i_{\text{pipe}}(t)$ at $t \approx 2 \times 10^{-5}$ s, the relative percentage difference between the two solutions was 0.41 %. Similar values were obtained for all the other considered conductors (including the OGWs and the soil, not shown in Figure 6).

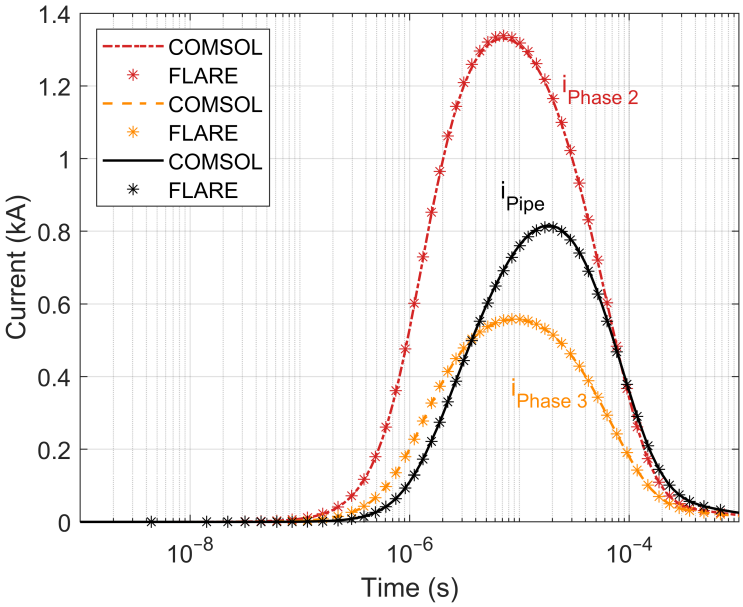

Figure 6. Comparison between the time-stepping solutions obtained with FLARE and COMSOL Multiphysics® for the same case of Figure 5 and the geometry of Figure 3.

3.4. COMSOL Simulation Settings

In order to validate the results given by FLARE, two different COMSOL Multiphysics® simulations were conducted considering both the above-described three-phase balanced system and lightning scenarios. Both simulations were performed using the AC/DC module and, specifically, the magnetic field (mf) submodule. In order to perform a fair comparison between FLARE and COMSOL Multiphysics®, we needed to employ the same triangular mesh for the two sets of calculations. Hence, we generated the triangular mesh using the `Gmsh` software, and then imported it into COMSOL Multiphysics® through an intermediate conversion to `.nas` format. In this regard, we report that the originally developed mesh led to some errors during the COMSOL Multiphysics® import process; the errors (misinterpreted triangles) were located in the fine-structured regions used for the pipeline. The errors disappeared once the local mesh size had been slightly relaxed. This fact did not seem to affect the results of the simulations, due to the relatively low frequency (and consequently the large penetration depth) used for both the time-harmonic analysis and the lightning simulation. This difference in local mesh size could, however, result in a meaningful alteration of the results for different physical situations involving high frequencies, e.g., calculations of impulse responses (not performed in this work with COMSOL Multiphysics®). Concerning the three-phase balanced system, a time-harmonic simulation was conducted, considering a frequency of 60 Hz and imposing the complex values of the three currents corresponding to the phase conductors. Each current had a magnitude of 250 A and a phase of 0, −120, and −240 degrees respectively. Note that, as stated previously and in order to avoid the influence of the self and mutually induced currents, the electrical conductivity of each of the three phase conductors was considered to be 0. Concerning the case of lightning-induced currents, a time-marching simulation was carried out, based on a backward difference formula scheme with variable accuracy order and adaptive Δt. In this case, the time-dependent current given by (1) was imposed

in phase conductor 1, which, for the reason stated above, has null conductivity. In order to calculate the values of the currents induced in the other phase conductors, the conductivity of the latter was considered to be 3.5×10^7 S/m.

3.5. Numerical Validation of the Inverse-Laplace Approach

As anticipated, FLARE also allows one to study transient phenomena through an inverse-Laplace approach. Specifically, a numerical inversion routine is used to call the FEM solver for a series of values of the (complex) variables s in (12) and (13). A complex MVP formulation similar to the one employed for time-harmonic problems is used to perform the finite element analysis in this case. The purely complex factor $j\omega$ used in the time-harmonic solver is now substituted by s, that may have a nonzero real part.

The aim of this section is twofold. First, we want to show how this approach can be applied to efficiently solve the same transient problem considered in the last section. Second, we want to provide verification of the numerical accuracy granted by this inverse-Laplace approach. Without losing any generality, we now focus solely on $i_{\text{pipe}}(t)$, for the sake of brevity. The most obvious way to obtain $i_{\text{pipe}}(t)$ using the inverse-Laplace approach would be to Laplace-transform the known current on phase 1, $i_{\text{phase 1}}(s) = \mathcal{L}\left\{i_{\text{phase 1}}(t)\right\}$, and subsequently use $i_{\text{phase 1}}(s)$ as the forcing term for the FEM solver in the s domain. In this way, $i_{\text{pipe}}(s)$ is obtained, and one can use one of the provided algorithms to perform the inverse transform $\mathcal{L}^{-1}\{i_{\text{pipe}}(s)\} = i_{\text{pipe}}(t)$. Unfortunately, in this way, the calculation must be repeated for every change in $i_{\text{phase 1}}(t)$.

Due to this, we instead opt for using the above described procedure to inverse-transform the pipeline electrical response to the unit impulse (The unit step $1/s$ can also be used alternatively, knowing that the unit impulse is just the first-derivative of the unit step). In this case, the forcing term for the FEM solver is just $\mathcal{L}\{\delta\} = 1$ and the output $h_{\text{pipe}}(t)$ of the inverse-Laplace routine is the time-domain pipeline current when the phase conductor is subjected to a unit current impulse. Now the time response of the pipeline to the generic current $g_{\text{phase 1}}(t)$ applied to phase conductor 1 can be obtained using the convolution theorem:

$$i_{\text{pipe}}(t) = h_{\text{pipe}}(t) * g_{\text{phase 1}}(t) = \int_0^t h_{\text{pipe}}(\tau) g_{\text{phase 1}}(t-\tau)\, d\tau. \tag{21}$$

The short notation $*$ is used to identify the convolution integral between two generic time-domain functions from here onward.

Three different numerical Laplace inversion algorithms are currently implemented in FLARE. The adaptive approach by D'Amore and colleagues [34] is used to compute the time-domain impulse response of the system, i.e., $h_{\text{pipe}}(t)$. We emphasize that $h_{\text{pipe}}(t)$ (output of the inversion routine) is the pipeline current response when phase conductor 1 is excited by a unit current impulse. Then, by setting $g_{\text{phase 1}}(t) = i_{\text{phase 1}}(t)$ with the expression of (20), we can find the time-domain pipeline current as a simple convolution.

As anticipated, the computational advantage of this approach is that, once $h_{\text{pipe}}(t)$ has been computed for the given geometry, one can easily find the pipeline response to any given current on phase 1 by updating $g_{\text{phase 1}}(t)$. More details on the convolution routine are provided in Section 3.6.

The default MATLAB® function `w=conv(u,v)` assumes that the two arrays `u` and `v` are defined at the same time-instants, which must necessarily be regularly spaced. In other words, the `conv` function is not particularly practical for working with variables known at unevenly spaced time-instants, such as $h_{\text{pipe}}(t)$ (computing the impulse response over an evenly spaced time grid would result in a prohibitive number of time steps, even with an implicit time discretization such as the one used in this work). For this reason we provide our own implementation of a convolution integral routine, `iw=intconv(tiw,tu,u,tv,v)`. The `intconv` function requires as input the array of the two signals to be combined, `u` and `v`, together with their respective temporal indices `tu` and `tv`. The two input signals are

interpolated internally and the convolution integral `iw` is then returned at times specified with the array `tiw`.

In order to validate this proposed methodology, we compared the pipeline current computed with this approach to the solution of the implicit time-stepping algorithm discussed in the previous section. The current impulse response of the pipeline $h_{\text{pipe}}(t)$ was computed over 250 instants, logarithmically spaced between 1×10^{-11} s and 10 s. Figure 7 shows the time-stepping results (black continuous line) compared to a subset of the values obtained using the convolution integral. The agreement between the two approaches was satisfactory over the whole time interval covered by the simulation. Focusing on the region with the largest discrepancy between the two approaches, highlighted in Figure 7, the pipeline current values yielded by the two approaches were 811.899 A (convolution) and 811.203 A (time domain). An even closer agreement could be obtained by employing a finer time grid for the pipeline impulse response calculation with the numerical inverse-Laplace approach.

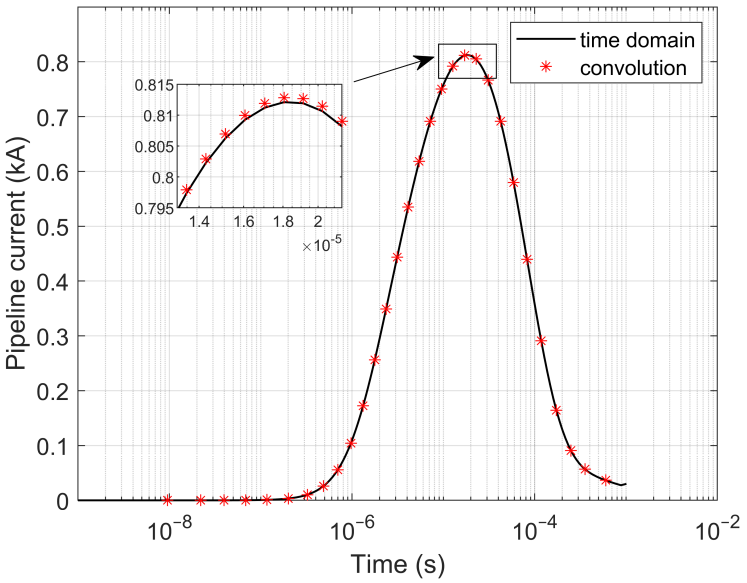

Figure 7. Comparison between the pipeline current computed with the implicit time-marching algorithm (time domain) and inverse-Laplace approaches (convolution) in FLARE. We denote the results of the inverse-Laplace as *convolution*, since the pipeline current is obtained by convolving the inverse-Laplace transform of the impulse response of the system and the lightning-induced current on phase 1.

We now want to briefly comment on the efficiency of the two approaches used in this work for transient problems, i.e., time-marching and the discussed inverse-Laplace technique. For the specific transient problem considered in this work, the time-marching analysis in FLARE (with a variable time step) required computation times between one and two minutes (depending on the initial time step length and sensitivity of the adaptive time step control). Comparing this performance to the ones of the inverse-Laplace approach is not trivial and is strongly dependent on the specific use case. The picture also changes depending on how the inverse-Laplace approach is used, i.e., if (1) the considered variable is directly anti-transformed or if (2) the inverse-Laplace approach is used to compute an impulse response of the system, to be later convoluted with a given signal. The number of function evaluations in the Laplace domain (i.e., solutions of a stationary problem for a single value of s) required to anti-transform the solution at a given time t may change considerably (depending on the source terms and the specific value of t) when using the adaptive procedure by D'Amore et al. Therefore, while 10 or 20 computations might be sufficient for a given $t = \tau_1$, up to 100 or 200 evaluations might be required for $t = \tau_2 \gg \tau_1$. Still, as we will see in a later section, a totally different situation is obtained when using

the Talbot and Euler algorithms. In any case, one of the advantages of the inverse-Laplace approach is that the problem can be massively parallelized—as we implemented in FLARE—given that the results at different time instants are independent. As a rule of thumb, if the inverse-Laplace is directly used without computing an impulse response (approach (1)), we can say that the time-marching approach is generally more efficient if used to observe the behavior of a quantity over large time-spans using a correspondingly large number of sampling points. In this case, the inverse-Laplace approach can still be competitive if the solution is just needed for a few key instants that can be computed in parallel. The situation changes if—as we implemented in this work—the above-described approach (2) is used. For the configuration in this work, obtaining a detailed impulse response (250 samples) took ≈1 h of wall-clock time. This relatively long operation only has to be carried out once for a given geometry, and the response to any given signal can be obtained through a convolution integral, requiring ≈10–100 ms. A final consideration about accuracy: truncation (and round-off) errors propagate over time during a time-marching simulation, and the truncation error magnitude depends on the time step length. This is not the case when using the inverse-Laplace approach, in that the solution at each instant is computed independently.

3.6. Comparison between Different Numerical Inversion Routines

As previously stated, FLARE includes three different algorithms for the numerical inversion of the Laplace-transformed problem. The first algorithm is made available by wrapping a MATLAB® function developed by D'Amore and colleagues [40]. This algorithm is based on a Fourier-series expansion of the function $f(t)$ to be transformed [34]. This expansion is then discretized using a trapezoidal rule with step size π/T to obtain an approximation of $f(t)$:

$$\tilde{f}(t) = \frac{\exp(\gamma t)}{T} \operatorname{Re}\left[\frac{F(\gamma)}{2} + \sum_{k=1}^{N} F\left(\gamma + \frac{ik\pi}{T}\right) \exp\left(\frac{ik\pi t}{T}\right)\right], \tag{22}$$

One of the main strengths of this algorithm is that it allows one to specify the minimum accuracy of the inverse transformation. In other words, the approximated values of $f(t)$ are bound to a user-defined relative tolerance:

$$\mathrm{TOL} \geq \left|\frac{f(t) - \tilde{f}(t)}{\tilde{f}(t)}\right| \tag{23}$$

We add two additional numerical inverse Laplace routines, implementing the Fourier expansion with Euler summation and Talbot algorithms. The MATLAB® implementations of the former two follow the framework introduced in [35], where an approximation of $\mathcal{L}^{-1}\{s\} = f(t)$, i.e., $\tilde{f}(t)$, is expressed as a linear combination of values of $F(s)$:

$$\tilde{f}(t) = \frac{1}{t} \sum_{k=0}^{n} \omega_k F\left(\frac{\alpha_k}{t}\right). \tag{24}$$

The complex quantities α_k and ω_k are known as nodes and weights, respectively. Notably, these only depend on n, i.e., the number of evaluations of $F(s)$ used to perform the inverse transform. This feature makes algorithms based on (24) very efficient for situations where the inverse transformation must be computed for multiple instants, such as the problems considered in this work.

Following the framework of (24), the Euler algorithm is formulated as

$$\tilde{f}(t) = \frac{10^{M/3}}{t} \sum_{k=0}^{2M} \eta_k \operatorname{Re}\left[F\left(\frac{\beta_k}{t}\right)\right], \tag{25}$$

while the Talbot algorithm is given by:

$$\tilde{f}(t) = \frac{2}{5t} \sum_{k=0}^{M-1} \mathrm{Re}\left[\gamma_k F\left(\frac{\delta_k}{t}\right)\right]. \tag{26}$$

We refer the reader to [35] for further details on the two expressions above.

Theoretically, $\tilde{f}(t)$ becomes more accurate the greater the value of n, thanks to a lower truncation error. In practice, the weights in (25) and (26) increase in magnitude with n, while oscillating in sign. This leads to round-off errors [53] for a constant system precision. In summary, the accuracy of the obtained solution only increases with greater values of n up to a certain optimum value, beyond which the ill-conditioning of the problem combines with numerical error to increase the total error [54].

Since FLARE is currently limited to double-precision calculations, we did not include the Gaver–Stehfest algorithm—also considered in [35]—since it requires a higher precision with respect to the Euler and Talbot algorithms. Both routines produce approximately $0.6M$ digits of accuracy for $\tilde{f}(t)$ with respect to the true solution $f(t)$, meaning that:

$$\left|\frac{f(t) - \tilde{f}(t)}{\tilde{f}(t)}\right| \approx 10^{-0.60M} \tag{27}$$

In other words, if the $\tilde{f}(t)$ must be accurate to the 6th digit with respect to the real solution $f(t)$, $M = 10$ must be used when performing the inverse-transform.

The positive integer M appearing in the above expression is related to the number of function evaluations n; specifically, $n = 2M + 1$ for the Euler algorithm and $n = M$ for the Talbot one. Both algorithms require a system precision $\approx M$, meaning that values of M above 16 are not recommended for double precision.

We compared the results yielded by the three algorithms described so far for a simple test case. Considering the same geometry as for the previous section, we computed the pipeline current $i_{\mathrm{pipe}}(t)$ when phase 1 of the transmission line was subject to a unity impulse voltage. The employed settings were the same as employed for the previously discussed case featuring the response to a current impulse on the same conductor, the only difference being $\sigma_{\mathrm{phase\ 1}} \neq 0$ for the present simulation. We selected three time instants for the comparison, corresponding to an early stage of the response ($\tau_1 = 2.63665 \times 10^{-9}$ s), when the pipeline current was close to its maximum absolute value ($\tau_2 = 4.54878 \times 10^{-5}$ s) and at a latter stage of the response ($\tau_3 = 0.194748$ s).

The results of the comparison are reported in Table 2. We took the results yielded by the algorithm proposed by D'Amore et al., and verified in the former sections, as the reference for this comparison. The pipeline current values computed with the above-described implementations of the Talbot and Euler techniques were collected for several different values of the parameter M. Instead of directly reporting the computed currents, we directly computed the percentage difference with respect to the reference current values, meaning that, e.g., using the Talbot algorithm with $M = 4$ we had $\left|\frac{i_{\mathrm{D'Amore}} - i_{\mathrm{Talbot}}}{i_{\mathrm{D'Amore}}}\right| \times 100 = 1.60 \times 10^{-2}$. For both algorithms the obtained accuracy was low or very low for values of M close to the upper or lower investigated limit. For $M = 4$, the round-off error was negligible, but from (27) only $\approx$2 digits of accuracy could be expected theoretically. The result was thus dominated by the truncation error. The large errors for values of M larger than the machine precision ($\approx$16) were instead due to round-off errors for the reasons discussed above and—in greater detail—in [53]. Not surprisingly, the most accurate results were obtained for both algorithms for $M = 8$ and $M = 16$. We note, however, that the Talbot algorithm seemed to consistently perform better than the Euler algorithm over the observed results. This was also true for the number of required function evaluations n, where the Talbot strategy required $n - 1$ less evaluations for the given value of M. Overall, for $M = 8$ and $M = 16$ the two techniques yield results in close or very close agreement with the

reference values, for a considerably lower number of function evaluations compared to the technique by D'Amore et al.

Table 2. Percentage error of the Talbot and Euler inverse-Laplace algorithms with respect to the pipeline current computed with the algorithm by D'Amore et al. for three time instants. M is the integer parameter used to control the Talbot and Euler algorithms. It relates to the number of function evaluations n and, approximately, to the required machine precision $\approx M$ and achieved accuracy through (27). The number of function evaluations required by the algorithm of D'Amore et al. is $n_{\tau_1} = 31$, $n_{\tau_2} = 235$, $n_{\tau_3} = 1333$.

Algorithm	**Settings**		**Time (s)**		
			$\tau_1 = 2.64 \times 10^{-9}$	$\tau_2 = 4.55 \times 10^{-5}$	$\tau_3 = 0.19$
				pipeline current (A)	
D'Amore et al.	Relative tolerance	1.00×10^{-5}	$-9.419\,87 \times 10^{2}$	$-7.274\,06 \times 10^{4}$	$3.076\,32 \times 10^{3}$
	M	n	% Error with respect to D'Amore et al.		
Talbot	4	4	1.60×10^{-2}	3.87×10^{-2}	1.23×10^{1}
	8	8	1.45×10^{-5}	3.75×10^{-6}	7.59×10^{-3}
	16	16	4.18×10^{-11}	1.10×10^{-8}	1.69×10^{-4}
	32	32	3.33×10^{-12}	1.29×10^{-7}	2.50×10^{-3}
	64	64	1.06×10^{-6}	1.77×10^{-2}	3.26×10^{2}
Euler	4	9	6.05×10^{-1}	2.67×10^{-1}	1.26
	8	17	1.27×10^{-3}	6.54×10^{-4}	5.12×10^{-3}
	16	33	5.64×10^{-9}	2.89×10^{-8}	4.21×10^{-4}
	32	65	9.50×10^{-7}	5.89×10^{-4}	1.15×10^{2}
	64	129	3.93×10^{2}	5.76×10^{7}	7.57×10^{11}

4. Conclusions

In this work, we introduced FLARE, an open-source MATLAB® code developed by the authors for the finite element analysis of electromagnetic interference on earth-return conductors due to the presence of high-voltage transmission lines. We considered a single section of a corridor with a buried steel pipeline underneath a typical 500 kV transmission line and computed the induced current on the pipeline under both sinusoidal steady-state and transient operating conditions of the HVAC line. We described an efficient vectorized implementation of a stationary complex solver and a transient solver with automatic time-stepping in FLARE and verified both of the obtained solutions by implementing the same problem in COMSOL Multiphysics®, using the AC/DC Module. The results yielded by FLARE and COMSOL Multiphysics® were compared in two different cases, i.e., a time-harmonic and a transient simulation. For the 60 Hz time-harmonic problem, the maximum percentage difference between the current magnitudes yielded by the two codes was ≈0.1% (pipeline). The maximum percentage phase difference was ≈0.21% (OGW 2). Considering the transient problem, we compared the percentage difference between the two maximum values of pipeline current, which was about ≈0.41%. Regarding the computation time, FLARE was faster by a factor of ≈1.4 for the considered time-harmonic problem, where the linear system stemming from the finite element discretization was only solved once. Conversely, COMSOL Multiphysics® was faster by a factor of ≈2 in the transient problem (note that a higher-order backward difference formula is employed by the COMSOL Multiphysics® time-marching solver, limiting the number of time steps employed). Subsequently, we discussed an alternative technique for the analysis of transient problems, based on the Laplace transform. This entails solving the problem in the complex domain for multiple values of the complex variable s and performing a numerical inverse-Laplace transformation to obtain the time-domain solution. We demonstrated this approach by anti-transforming the pipeline response to a current impulse applied to one of the transmission line phase conductors. The transient response of

the pipeline for any given phase current can be then obtained through a simple convolution integral, with obvious performance advantages over time-marching methods. We compared the pipeline current obtained through time-marching with the one obtained from the described inverse-Laplace approach, obtaining a full agreement. Finally, we compared the accuracy and computational efficiency of the three different inverse-Laplace algorithms implemented in FLARE, providing general recommendations for their use in the context of double-precision environments. In future works, FLARE will be further developed by adding the possibility of using the inverse-Laplace approach when dealing with multiple sections of a corridor. This will allow performing quasi-3D simulations, which can take into account imperfect earthings of the conductors and, more generally, currents that do not flow in the direction perpendicular to the given section.

Author Contributions: Conceptualization, A.P. and A.C.; Methodology, A.P. and A.C.; Software, A.P.; Validation, A.P. and G.P.; Formal analysis, A.P. and A.C.; Investigation, A.P.; Data curation, A.P.; Writing—original draft, A.P. and G.P.; Writing—review & editing, A.P., F.R. and L.S.; Supervision, A.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lucca, G. AC interference from a faulty power line on nearby buried pipelines: Influence of the surface layer soil. *IET Sci. Meas. Technol.* **2020**, *14*, 225–232. [CrossRef]
2. Zhang, Y.; Weng, W.G. Bayesian network model for buried gas pipeline failure analysis caused by corrosion and external interference. *Reliab. Eng. Syst. Saf.* **2020**, *203*, 107089. [CrossRef]
3. Cetin, O.; Duzkaya, H.; Taplamacioglu, M.C. Analysis of Transmission Line Electromagnetic Interference on Touch and Step Voltages on Buried Gas Pipeline under Different Shielding and Resistivity Conditions. In Proceedings of the 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 1–3 July 2021; pp. 1–4. [CrossRef]
4. Brenna, A.; Beretta, S.; Ormellese, M. AC Corrosion of Carbon Steel under Cathodic Protection Condition: Assessment, Criteria and Mechanism. A Review. *Materials* **2020**, *13*, 2158. [CrossRef]
5. Al-Gabalawy, M.; Mostafa, M.A.; Hamza, A.S. Implementation of different intelligent controllers for mitigating the AC corrosion of metallic pipelines considering all HVOHTLs operation conditions. *ISA Trans.* **2021**, *117*, 251–273. [CrossRef]
6. Muresan, A.; Papadopoulos, T.A.; Czumbil, L.; Chrysochos, A.I.; Farkas, T.; Chioran, D. Numerical Modeling Assessment of Electromagnetic Interference Between Power Lines and Metallic Pipelines: A Case Study. In Proceedings of the 2021 9th International Conference on Modern Power Systems (MPS), Cluj-Napoca, Romania, 16–17 June 2021; pp. 1–6. [CrossRef]
7. Luo, Y.; Lin, N.; Zhou, S.; Li, S.; Wang, H. Effects of electromagnetic interference and crevice on corrosion of natural gas pipelines. *IOP Conf. Ser. Earth Environ. Sci.* **2021**, *675*, 012061. [CrossRef]
8. Liu, H.; Li, Q.; Li, H.; Xue, Z.; Han, M.; Liu, L. Characteristics and effects of electromagnetic interference from UHVDC and geomagnetic storms on buried pipelines. *Int. J. Electr. Power Energy Syst.* **2021**, *125*, 106494. [CrossRef]
9. Charalambous, C.A.; Nikolaidis, A.I. Complete Method to Assess the DC Corrosion Impact on Pipeline Systems During the Planning and Approval Stages of HVDC Systems With Earth Current Return. *IEEE Access* **2022**, *10*, 127550–127562. [CrossRef]
10. Li, Y.; Liu, L.; Ya, S.; Long, Y.; Li, K.; Cao, X. Research on the Protection Principle and Influencing Factors of Zinc Strip on Buried Pipelines around AC Subway. In Proceedings of the 2022 IEEE International Conference on High Voltage Engineering and Applications (ICHVE), Chongqing, China, 25–29 September 2022; pp. 1–4. [CrossRef]
11. Ma, C.; Liu, C. Influence of Pipeline Insulation Leakage Points on the Distribution of Geomagnetically Induced Current and Pipe-Soil Potential. *IEEE Access* **2019**, *7*, 147470–147480. [CrossRef]
12. CIGRE. *Guide on the Influence of High Voltage AC Power Systems on Metallic Pipelines*; Technical Report; Cigré Working Group 36.02: Paris, France, 1995.
13. Taflove, A.; Dabkowski, J. Prediction Method for Buried Pipeline Voltages Due to 60 Hz AC Inductive Coupling Part I—Analysis. *IEEE Trans. Power Appar. Syst.* **1979**, *PAS-98*, 780–787. [CrossRef]
14. Dabkowski, J.; Taflove, A. Prediction Method for Buried Pipeline Voltage Due to 60 Hz AC Inductive Coupling Part II—Field test Verification. *IEEE Trans. Power Appar. Syst.* **1979**, *PAS-98*, 788–794. [CrossRef]
15. Machczynski, W.; Budnik, K.; Szymenderski, J. Assessment of DC traction stray currents effects on nearby pipelines. *COMPEL Int. J. Comput. Math. Electr. Electron. Eng.* **2016**, *35*, 1468–1477. [CrossRef]

16. Cristofolini, A.; Popoli, A.; Sandrolini, L. A comparison between Carson's formulae and a 2D FEM approach for the evaluation of AC interference caused by overhead power lines on buried metallic pipelines. *Prog. Electromagn. Res.* **2017**, *79*, 39–48. [CrossRef]
17. Lucca, G. Influence of steel non-linearity in assessing 50–60 HZ interference on pipelines. *Prog. Electromagn. Res. M* **2018**, *74*, 1–10. [CrossRef]
18. Ametani, A.; Yoneda, T.; Baba, Y.; Nagaoka, N. An Investigation of Earth-Return Impedance Between Overhead and Underground Conductors and Its Approximation. *IEEE Trans. Electromagn. Compat.* **2009**, *51*, 860–867. [CrossRef]
19. Dawalibi, F.P.; Southey, R.D. Analysis of electrical interference from power lines to gas pipelines. I. Computation methods. *IEEE Trans. Power Deliv.* **1989**, *4*, 1840–1846. [CrossRef]
20. Christoforidis, G.C.; Labridis, D.P.; Dokopoulos, P.S. A hybrid method for calculating the inductive interference caused by faulted power lines to nearby buried pipelines. *IEEE Trans. Power Deliv.* **2005**, *20*, 1465–1473. [CrossRef]
21. Schoonjans, B.; Deconinck, J. Calculation of HVAC inductive coupling using a generalized BEM for Helmholtz equations in unbounded regions. *Int. J. Electr. Power Energy Syst.* **2017**, *84*, 242–251. [CrossRef]
22. Micu, D.D.; Christoforidis, G.C.; Czumbil, L. AC interference on pipelines due to double circuit power lines: A detailed study. *Electr. Power Syst. Res.* **2013**, *103*, 1–8. [CrossRef]
23. Cristofolini, A.; Popoli, A.; Sandrolini, L. Numerical Modelling of Interference from AC Power Lines on Buried Metallic Pipelines in Presence of Mitigation Wires. In Proceedings of the 2018 IEEE International Conference on Environment and Electrical Engineering and 2018 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), Palermo, Italy, 12–15 June 2018; pp. 1–6. [CrossRef]
24. Popoli, A.; Sandrolini, L.; Cristofolini, A. Finite Element Analysis of Mitigation Measures for AC Interference on Buried Pipelines. In Proceedings of the 2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe), Genova, Italy, 11–14 June 2019; pp. 1–5. [CrossRef]
25. Popović, L.M. 7—Inductive influence of high-voltage and extra-high voltage lines on surrounding metal installations. In *Relevant Characteristics of Power Lines Passing through Urban Areas*; Popović, L.M., Ed.; Academic Press: Cambridge, MA, USA, 2022; pp. 151–189. [CrossRef]
26. Haynes, G.J.; Manning, T.; Baete, C.; Barton, L. Variances in pipeline AC interference computational modeling. In Proceedings of the Corrosion Conference and Expo 2019, Nashville, TN, USA, 24–28 March 2019.
27. Southey, R.D.; Dawalibi, F.P.; Vukonich, W. Recent advances in the mitigation of AC voltages occurring in pipelines located close to electric transmission lines. *IEEE Trans. Power Deliv.* **1994**, *9*, 1090–1097. [CrossRef]
28. Andolfato, R.; Cuccarollo, D.; Fara, I. *Electromagnetic Interferences between Power Systems and Pipelines. Field vs. Circuit Theory Based Models*; NACE International: Milano, Italy, 2021.
29. Christoforidis, G.C.; Labridis, D.P.; Dokopoulos, P.S. Inductive interference on pipelines buried in multilayer soil due to magnetic fields from nearby faulted power lines. *IEEE Trans. Electromagn. Compat.* **2005**, *47*, 254–262. [CrossRef]
30. Martins-Britto, A.G.; Moraes, C.M.; Lopes, F.V. Transient electromagnetic interferences between a power line and a pipeline due to a lightning discharge: An EMTP-based approach. *Electr. Power Syst. Res.* **2021**, *197*, 107321. [CrossRef]
31. Cristofolini, A.; Popoli, A.; Sandrolini, L.; Pierotti, G.; Simonazzi, M. Laplace transform for finite element analysis of electromagnetic interferences in underground metallic structures. *Appl. Sci.* **2022**, *12*, 872. [CrossRef]
32. Wen, T.; Cui, X.; Li, X.; Liu, S.; Zhao, Z. A finite element method for the transient electric field by using indirect Laplace transform with high accuracy. *CSEE J. Power Energy Syst.* **2022**, 1–11. [CrossRef]
33. Chen, H.T.; Chen, C.K. Hybrid Laplace transform/finite-element method for two-dimensional transient heat conduction. *J. Thermophys. Heat Transf.* **1988**, *2*, 31–36. [CrossRef]
34. D'Amore, L.; Laccetti, G.; Murli, A. Algorithm 796: A Fortran software package for the numerical inversion of the Laplace transform based on a Fourier series method. *ACM Trans. Math. Softw.* **1999**, *25*, 306–315. [CrossRef]
35. Abate, J.; Whitt, W. A Unified Framework for Numerically Inverting Laplace Transforms. *INFORMS J. Comput.* **2006**, *18*, 408–421. [CrossRef]
36. MATLAB. *R2022a*; The MathWorks Inc.: Natick, MA, USA, 2022.
37. Popoli, A.; Cristofolini, A.; Sandrolini, L.; Abe, B.T.; Jimoh, A. Assessment of AC interference caused by transmission lines on buried metallic pipelines using FEM. In Proceedings of the 2017 International Applied Computational Electromagnetics Society Symposium-Italy (ACES), Firenze, Italy, 26–30 March 2017; pp. 1–2. [CrossRef]
38. Mazumder, S. *Numerical Methods for Partial Differential Equations: Finite Difference and Finite Volume Methods*; Academic Press: Cambridge, MA, USA, 2015.
39. MATLAB Inc. *mldivide MATLAB Function*; MATLAB Inc.: Natick, MA, USA, USA, 2023.
40. D'Amore, L.; Laccetti, G.; Murli, A. An implementation of a Fourier series method for the numerical inversion of the Laplace transform. *ACM Trans. Math. Softw. (TOMS)* **1999**, *25*, 279–305. [CrossRef]
41. Cohen, A. *Numerical Methods for Laplace Transform Inversion*; Numerical Methods and Algorithms, Springer: Boston, MA, USA, 2007; Volume 5. [CrossRef]
42. de Hoog, F.R.; Knight, J.H.; Stokes, A.N. An Improved Method for Numerical Inversion of Laplace Transforms. *SIAM J. Sci. Stat. Comput.* **1982**, *3*, 357–366. [CrossRef]
43. Chen, L. Programming of Finite Element Methods in MATLAB. *arXiv* **2018**, arXiv:1804.05156. https://doi.org/10.48550/arXiv.1804.05156.

44. Rahman, T.; Valdman, J. Fast MATLAB assembly of FEM matrices in 2D and 3D: Nodal elements. *Appl. Math. Comput.* **2013**, *219*, 7151–7158. [CrossRef]
45. Cuvelier, F.; Japhet, C.; Scarella, G. An efficient way to assemble finite element matrices in vector languages. *BIT Numer. Math.* **2016**, *56*, 833–864. [CrossRef]
46. *COMSOL Multiphysics, 6.1*; COMSOL AB: Stockholm, Sweden, 2022.
47. Popoli, A.; Sandrolini, L.; Cristofolini, A. A quasi-3D approach for the assessment of induced AC interference on buried metallic pipelines. *Int. J. Electr. Power Energy Syst.* **2019**, *106*, 538–545. [CrossRef]
48. Wang, X.; Wang, Y.; Sun, T.; Yang, X.; Yang, L.; Qi, Y. Study of transmission line AC interference with steel-buried pipelines under lightning strikes. *Electr. Power Syst. Res.* **2023**, *218*, 109226. [CrossRef]
49. Yao, C.; Wu, H.; Mi, Y.; Ma, Y.; Shen, Y.; Wang, L. Finite difference time domain simulation of lightning transient electromagnetic fields on transmission lines. *IEEE Trans. Dielectr. Electr. Insul.* **2013**, *20*, 1239–1246. [CrossRef]
50. Qais, M.; Khaled, U. Evaluation of V–t characteristics caused by lightning strokes at different locations along transmission lines. *J. King Saud Univ. Eng. Sci.* **2018**, *30*, 150–160. [CrossRef]
51. Geuzaine, C.; Remacle, J.F. Gmsh: A three-dimensional finite element mesh generator with built-in pre-and post-processing facilities. In Proceedings of the Second Workshop on Grid Generation for Numerical Computations, Tetrahedron II, Le Chesnay, France, 17–19 October 2007.
52. Piantini, A. (Ed.) *Lightning Interaction with Power Systems. Volume 2: Applications*; Number 172 in IET Energy Engineering Series; The Institution of Engineering and Technology: London, UK, 2020.
53. Abate, J.; Valkó, P.P. Multi-precision Laplace transform inversion. *Int. J. Numer. Methods Eng.* **2004**, *60*, 979–993. [CrossRef]
54. Davies, B. *Integral Transforms and Their Applications*, 3rd ed.; Number 41 in Texts in Applied Mathematics; Springer: New York, NY, USA, 2002.