# Telegrapher's Eequations Solved with MFEM Library

Denis Lachapelle

February 2025

## 1 Introduction

This document explains the telegrapher's equations simulation using finite differences method; one first version using python (numpy and scipy) and a second version based on MFEM library in C++.

## 2 Theory, Finite Differences

### 2.1 Continuous Time Equations

The telegraph equations are:

$$\frac{\partial V}{\partial x} + L\frac{\partial I}{\partial t} + RI = 0 \tag{1}$$

$$\frac{\partial I}{\partial x} + C\frac{\partial V}{\partial t} + GV = 0 \tag{2}$$

### 2.2 Discretization

1. Assume the transmission line is divided in N segments (nbrSeg) with nodes 0 to N. There is N+1 nodes. So nodes are located at $n\Delta x$.
2. Time is sampled at $k\Delta T$.
3. Assume the $\frac{\partial V(t,x)}{\partial x}$ and $\frac{\partial I(t,x)}{\partial x}$ are constant over each segment.

So at a given time $k\Delta T$ we can express the $\frac{\partial V(t,x)}{\partial x}$ and $\frac{\partial I(t,x)}{\partial x}$ as difference $\frac{V(k\Delta T,(n+1)\Delta x)-V(k\Delta T,(n)\Delta x)}{\Delta x}$ and $\frac{I(k\Delta T,(n)\Delta x)-I(k\Delta T,(n-1)\Delta x)}{\Delta x}$.

Rewrite the equations 1 and 2 with the $\frac{\partial}{\partial t}$ on the left side.

$$\frac{\partial I}{\partial t} = -\frac{RI}{L} - \frac{1}{L}\frac{\partial V}{\partial x} \tag{3}$$

$$\frac{\partial V}{\partial t} = -\frac{GV}{C} - \frac{1}{C}\frac{\partial I}{\partial x} \tag{4}$$

We can write matrix equations....

$$\frac{\partial I}{\partial t} = \begin{bmatrix} Dv & Ri \end{bmatrix} \begin{bmatrix} V^k \\ I^k \end{bmatrix} \tag{5}$$

$$\frac{\partial V}{\partial t} = \begin{bmatrix} Gv & Di \end{bmatrix} \begin{bmatrix} V^k \\ I^k \end{bmatrix} \tag{6}$$

The two equations above can be written as a single matrix equation...

$$\begin{bmatrix} \frac{\partial V}{\partial t} \\ \frac{\partial I}{\partial t} \end{bmatrix} = \begin{bmatrix} GvDi \\ DvRi \end{bmatrix} \begin{bmatrix} V^k \\ I^k \end{bmatrix} \tag{7}$$

The Dv matrix ix nbrSeg x (nbrSeg+1) as example for nbrSeg=5 ...

$$\frac{-1}{Lh} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{8}$$

Notice the matrix is complete in the sense that the derivative is fully estimated for each row, this correspond to the fact that each current segment is enclosed by two voltage nodes.

The Di matrix ix (nbrSeg+1) x nbrSeg as example for nbrSeg=5 ...

$$\frac{-1}{Ch}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{9}$$

Notice the matrix is not complete in the sense that the derivative are not fully estimated for first and last row, this correspond to the fact that for the end voltage nodes there is no source and load current. That will have to be considered in the boundary conditions.

Gv is (nbrSeg+1) x (nbrSeg+1) identity matrix scale by either -G/C.

Ri is nbrSeg x nbrSeg identity matrix scale by either -R/L.

## 2.3 Adding a Source

To add the source with a series resistor we should write the equation for $\frac{\partial V_0}{\partial t}$ which one will replace the first row equations. The equation is the same as equation 4 with added current from Rs.

$$\frac{\partial V_0}{\partial t} = -\frac{GV_0}{C} - \frac{1}{C}\frac{\partial I}{\partial x} + \frac{Vs}{RsCh} - \frac{V_0}{RsCh} \tag{10}$$

This done by adding a voltage node Vs, a row to compute Vs.

- the first row is all zero since $\frac{\partial V_s}{\partial t} = 0$ and in fact is not used.
- the second row is 1/(Rs C h) -1/(Rs C h) 0 0 ... 0.
- the last sub matrix is all 0.0.

The resulting matrix equation is ...

$$\begin{bmatrix} Vs \\ \frac{\partial V}{\partial t} \\ \frac{\partial I}{\partial t} \end{bmatrix} = \left[ \begin{bmatrix} 0 & 0 & 0 \\ 0 & Gv & Di \\ 0 & Dv & Ri \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{RsC} & \frac{-1}{RsC} & 0 \\ 0 & 0 & 0 \end{bmatrix} \right] \begin{bmatrix} Vs \\ V^k \\ I^k \end{bmatrix} \tag{11}$$

## 2.4 Adding a Load

Now we need to add the right boundary which is a load Rl. The last voltage node will get influenced by Rl, the equation becomes ...

$$\frac{\partial V_N}{\partial t} = -\frac{GV_N}{C} - \frac{1}{C}\frac{\partial I_{N-1}}{\partial x} - \frac{V_N}{RlCh} \tag{12}$$

The resulting matrix equation is ...

$$\begin{bmatrix} Vs \\ \frac{\partial V}{\partial t} \\ \frac{\partial I}{\partial t} \end{bmatrix} = \left[ \begin{bmatrix} 0 & 0 & 0 \\ 0 & Gv & Di \\ 0 & Dv & Ri \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{RsCh} & \frac{-1}{RsCh} & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{-1}{RsCh} & 0 \\ 0 & 0 & 0 \end{bmatrix} \right] \begin{bmatrix} Vs \\ Vk \\ Ik \end{bmatrix} \tag{13}$$

The non zero value is at location (nbrSeg+1, nbrSeg+1).

## 2.5 Time Stepping

Using the above operator we will step in time using RK4.
In principle it produce something similar to this...

$$\begin{bmatrix} Vs \\ V^{k+1} \\ I^{k+1} \end{bmatrix} = \begin{bmatrix} V_s \\ V^k \\ I^k \end{bmatrix} + \Delta t \begin{bmatrix} 0 \\ \frac{\partial V}{\partial t} \\ \frac{\partial I}{\partial t} \end{bmatrix} \tag{14}$$

# 3 Python Implementation

The program is named stltfdrk4.py for Single Transmission Line Transient Finite Difference runge kutta 4.

The signal injected is gaussian pulse centered at 100ns with a Tau of 20ns multiplied by a triangular window of 200ns wide centered at 100ns.

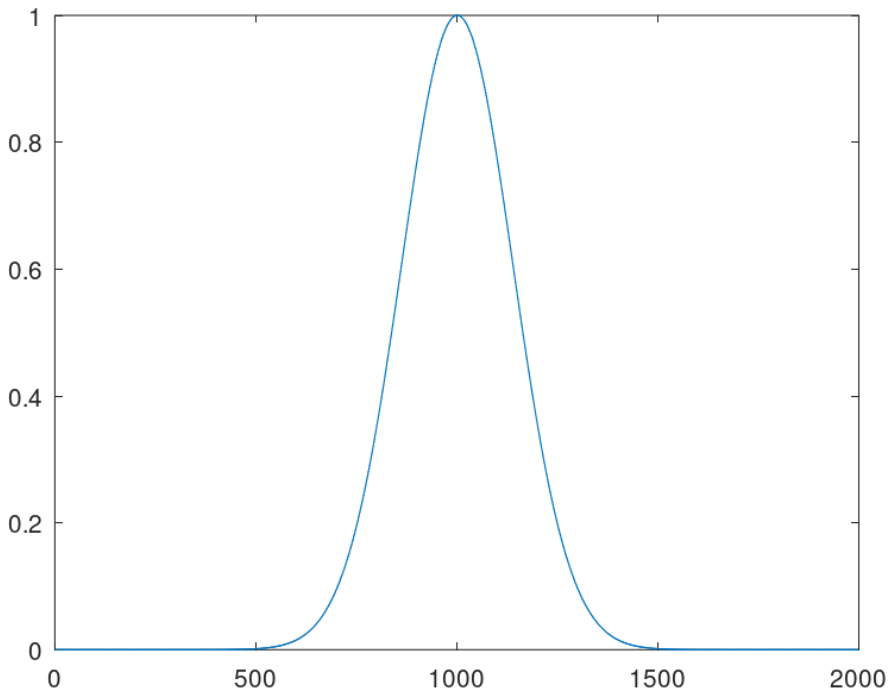


Figure 1: Gaussian Pulse

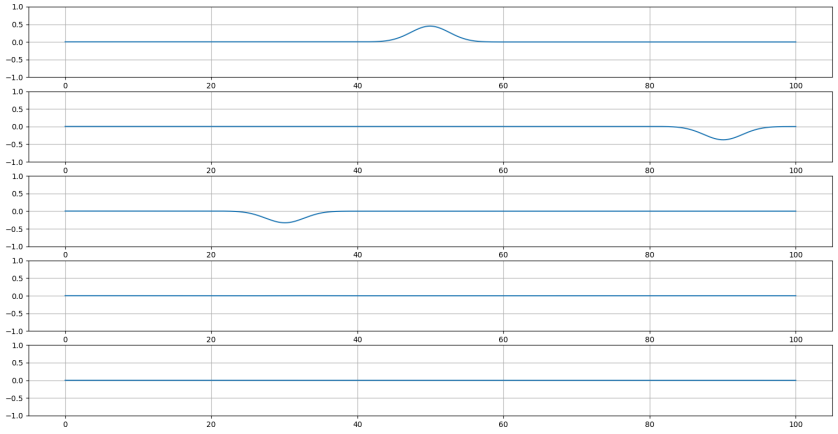


Figure 2: Test pulse propagating, Rs=50, Rl=1

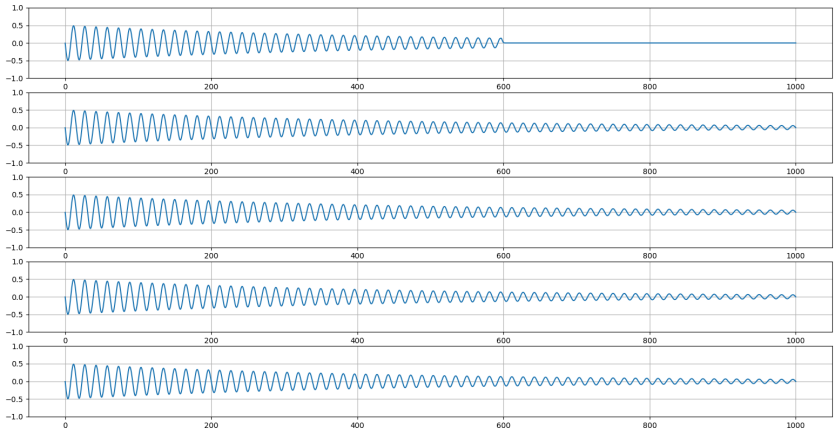


Figure 3: 13 MHz sinewave, coax=1000m, Rs=50, Rl=50.

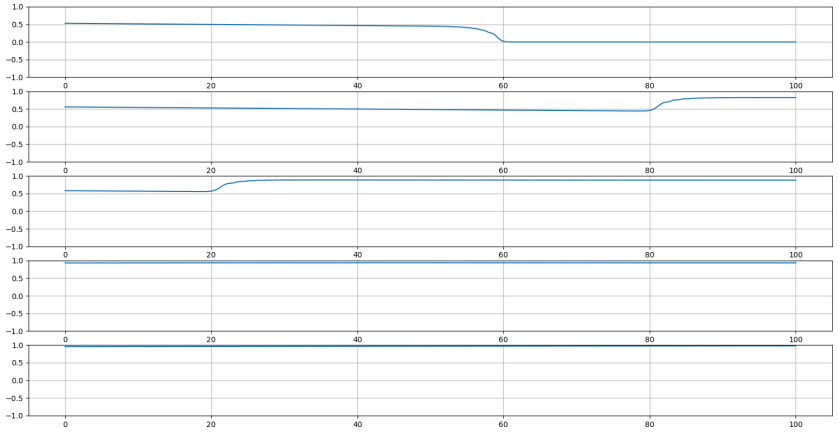


Figure 4: Step, cable 100m, Rs=50, Rl=open

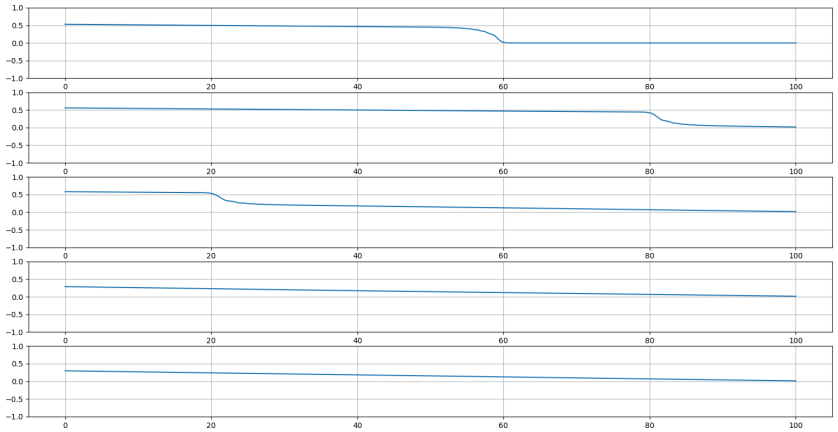


Figure 5: Step, cable 100m, Rs=50, Rl=50

# 4 MFEM Implementation

The program is named stltfdrk4.cpp for Single Transmission Line Transient Finite Difference runge kutta 4.

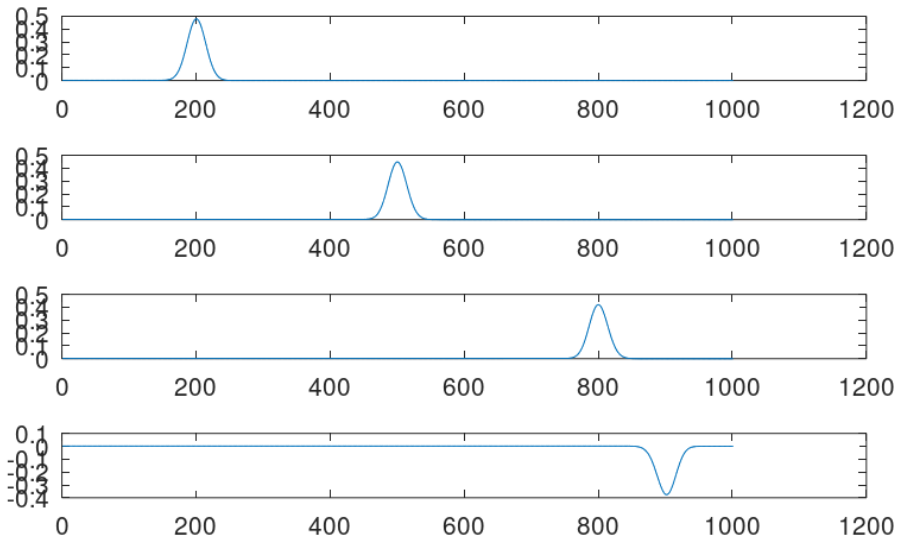


Figure 6: Test pulse propagating, Rs=50, Rl=1, MFEM

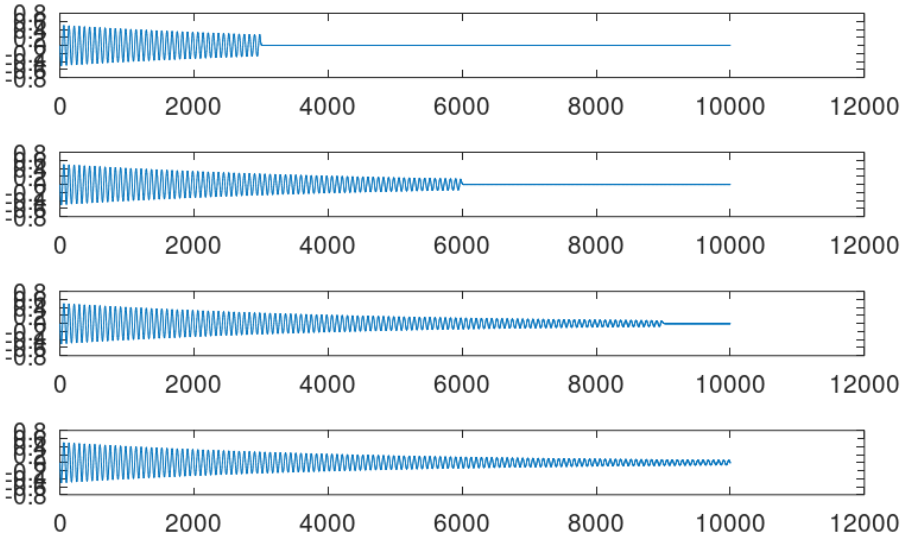


Figure 7: 13 MHz sinewave, coax=1000m, Rs=50, Rl=50, MFEM

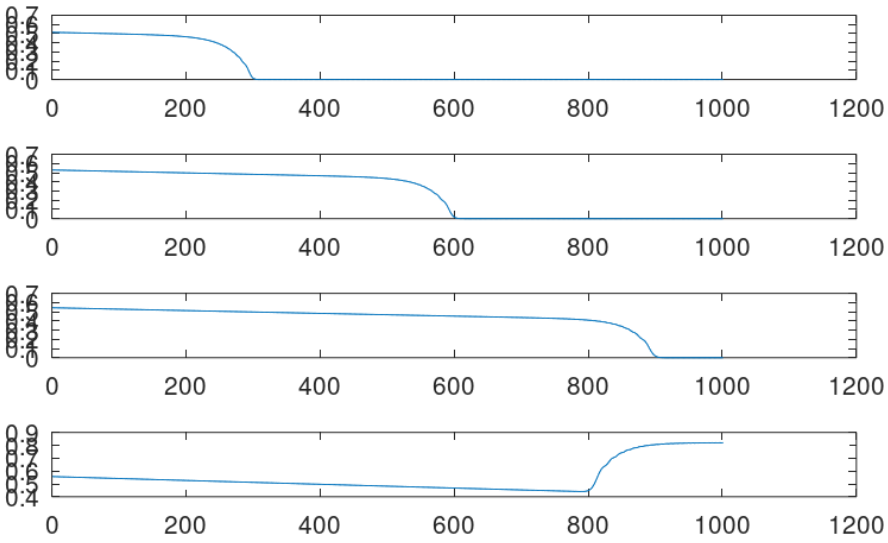


Figure 8: Step, cable 100m, Rs=50, Rl=open

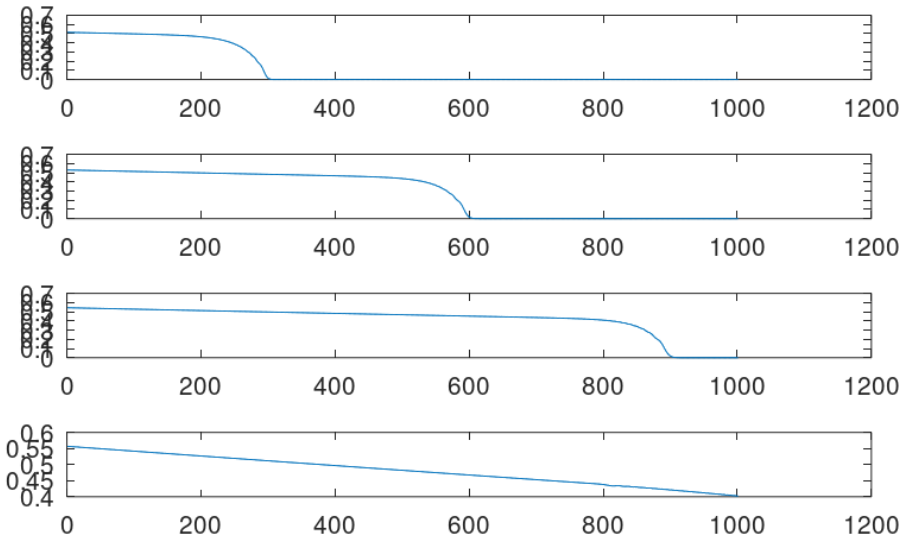


Figure 9: Step, cable 100m, Rs=50, Rl=50

# 5 Conclusion

The telegrapher's equations are resolved using finite difference method first in python using numpy and scipy and then C++ using MFEM library.