telegraph equations solved with MFEM library

Denis Lachapelle

February 2025

1 Introduction

This document explains the telegraph equations simulation using finite differences method; one first version using python and a second version based on MFEM library in C++ (not done yet).

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 TL is divided in N segments (nbrSeg) with node 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 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 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.

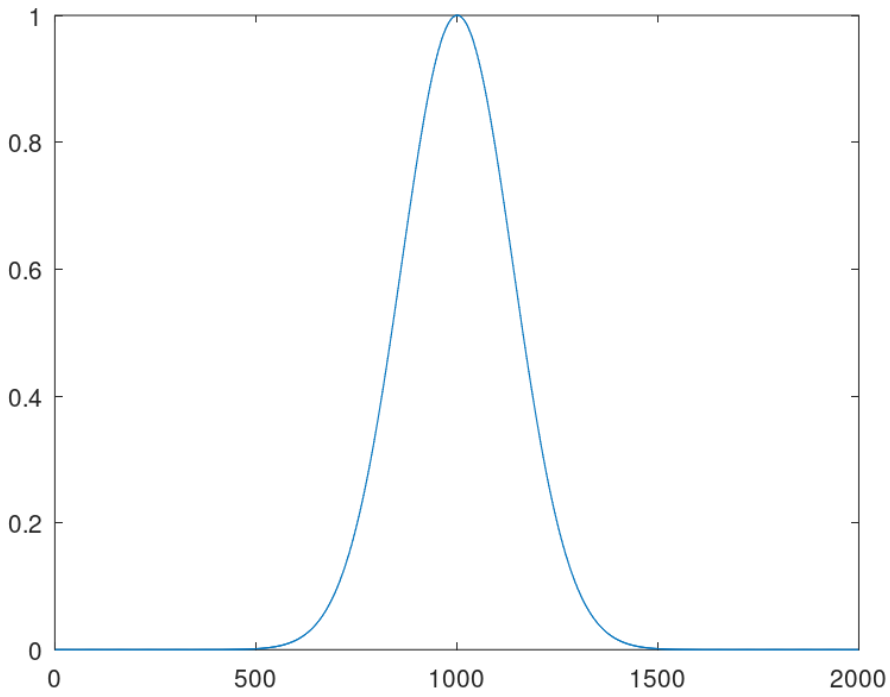


Figure 1: Gaussian Pulse

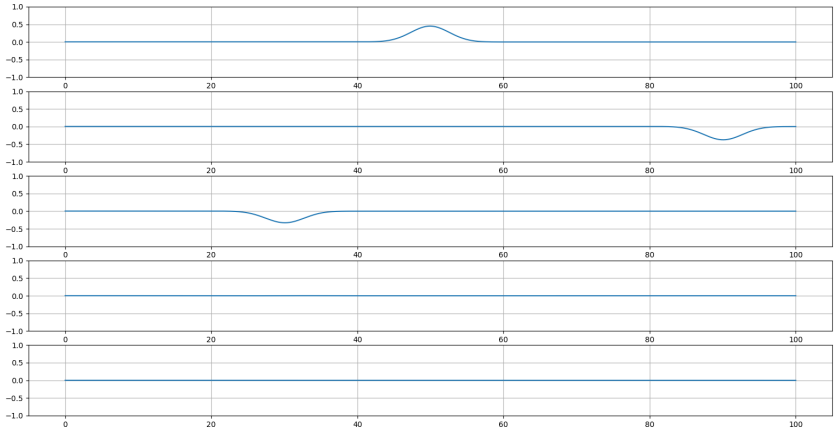


Figure 2: Test pulse propagating, Rs=50, Rl=1

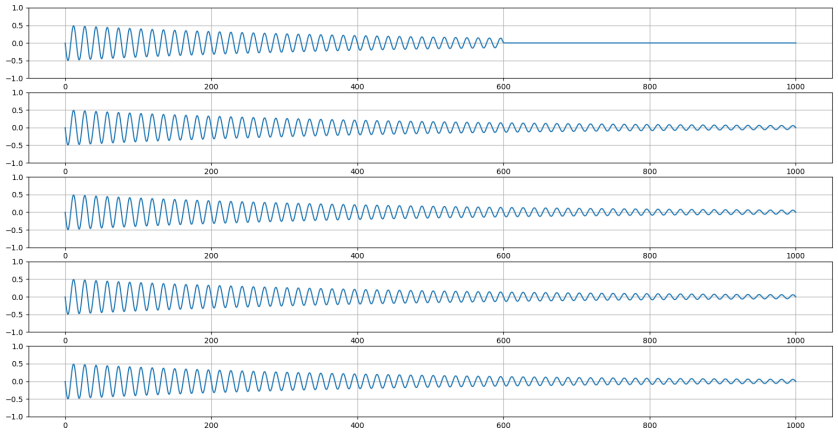


Figure 3: 13 MHz sinewave, coax=1000m, Rs=50, Rl=50.

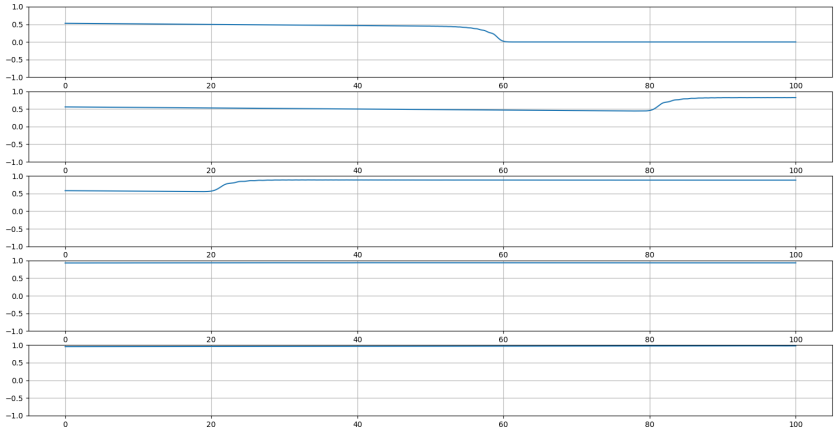


Figure 4: Step, cable 100m, Rs=50, Rl=open

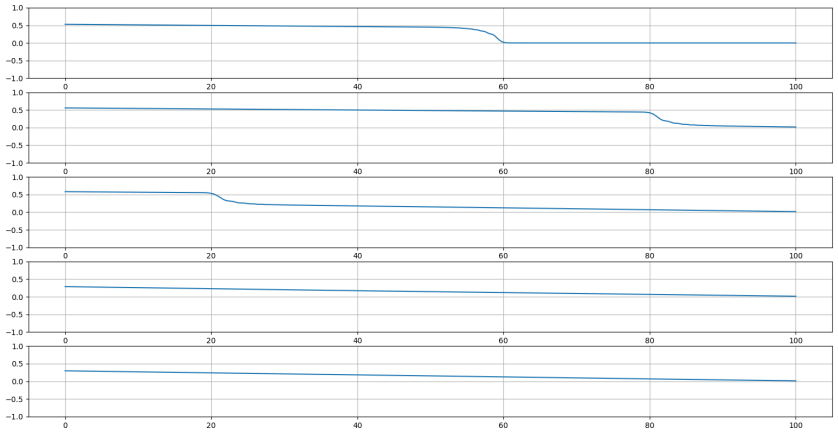


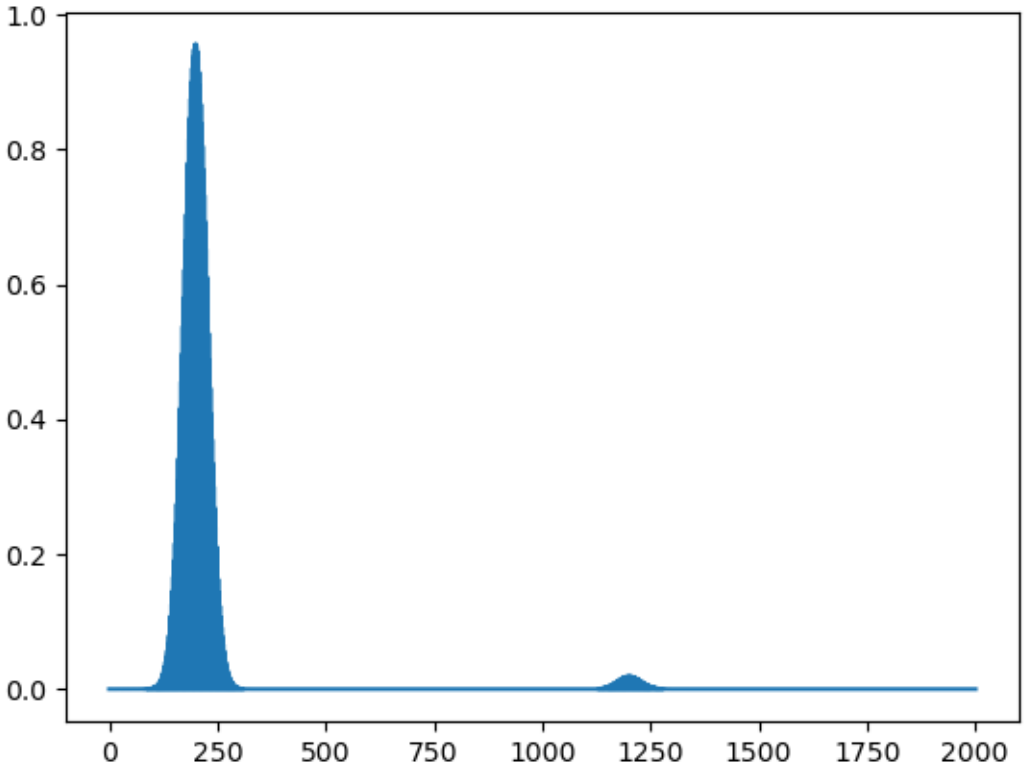
Figure 5: Step, cable 100m, Rs=50, Rl=

4 MFEM Implementation

DO NOT WORK YET.

The program is named stltfdrk4.cpp for Single Transmission Line Transient Finite Difference runge kutta 4.
The software do not work fine; I cant find any error. the next step...

- rewrite the software in python or octave.



ChatGPT: the resulting signal appear to be with a zero on all odd nodes.
ChatGPT said: Yes, if the resulting signal shows zeros on all odd nodes (e.g., nodes 1, 3, 5, ...), it's a strong hint that there's a structural symmetry or a bug in the way your system is discretized — particularly in how the derivative matrices Di and Dv are built.

The solution according to chatgpt is to have nbrSeg+1 voltage nodes and nbrSeg current in between voltage nodes. So the block matrix will be like this:

$$\begin{bmatrix} \frac{\partial V}{\partial t} \\ \frac{\partial I}{\partial t} \end{bmatrix} = \begin{bmatrix} GvDi \\ DvRi \end{bmatrix} \begin{bmatrix} V^k \\ I^k \end{bmatrix} \tag{15}$$

Gv dimensions will be nbrSeg+1 by nbrSeg+1.
Di dimensions will be nbrSeg+1 by nbrSeg.
Dv dimensions will be nbrSeg by nbrSeg+1.
Ri dimensions will be nbrSeg by nbrSeg.

So the block matrix dimensions are 2 nbrSeg+1 by 2 nbrSeg+1.

$$\begin{bmatrix} N+1, N+1 & N+1, N \\ N, N+1 & N, N \end{bmatrix} \tag{16}$$

The following figure shows the pulse propagating along a 100m RG-58 transmission line.

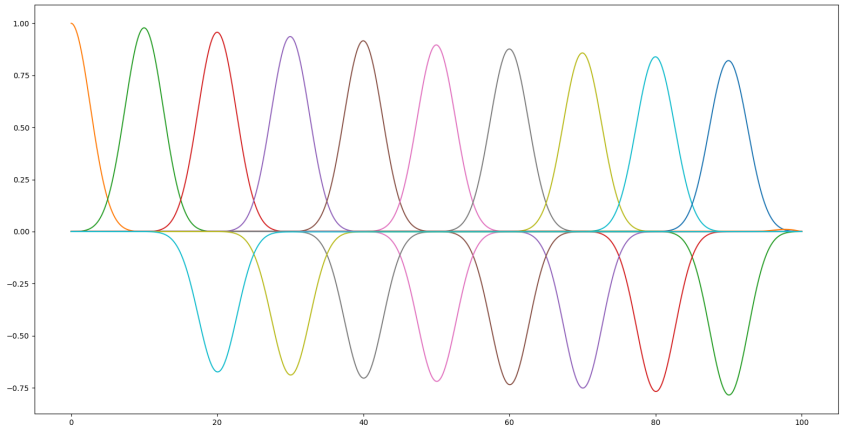


Figure 6: Test pulse propagating

The simulation works fine using Euler Forward difference to differentiate v(x) and I(x) against x and Runge Kutta 4 for time stepping.

The reflected voltage is inverted, this looks like the end of the transmission line is shorted, this is the case because the last row is all zero, so di/dx cannot

be ... THAT WILL BE TO COMPLETE.

Consider inserting this note somewhere... The block matrix take care of the transmission line and the following code lines take care of the left and right boundary conditions.

```
#apply the left boundary condition which is a voltage source with Rs in séries.
x[0, 0] = x[0, 0] + deltaT*(SourceFunction(Time) - x[0, 0])/(Rs*C*h)

#apply the right boundary condition whichis a load Rl.
x[nbrSeg, 0] = x[nbrSeg, 0] - deltaT*x[nbrSeg, 0]/(Rl*C*h)
```

But this method makes the RK4 not applied to the boundary. So the best way would be to include the boundary conditions within the block matrix.