Project Applied Electromagnetism: FDTD Simulation of Lossless Transmission Lines

Prof. Dries Vande Ginste*[1], ir. Pieter Decleer†[1], and ir. Dries Bosman‡[1]

[1]Electromagnetics Group, Department of Information Technology, Ghent University, Technologiepark-Zwijnaarde 126, 9052 Gent, Belgium.

October 25, 2020

1 Introduction

The purpose of this project is to numerically simulate the behavior of voltages and currents on a transmission line as a function of time. To this end, a well-known numerical technique, i.e., the Finite-Difference Time-Domain (FDTD) method, will be used. At present, many numerical solvers for Maxwell's equations based on FDTD (in three dimensions) are commercially available, e.g., CST Microwave Studio (see www.cst.com) and the FDTD-solver which is part of Keysight's EMPro (see http://www.keysight.com/en/pc-1297143/empro-3d-em-simulation-software).
The FDTD method relies on a leapfrog scheme both in space and in time, as further explained in this document. Special attention will be devoted to the correct inclusion of the Thévenin source driving the transmission line and to the load impedance.

2 Basic problem description

The transmission line under investigation is depicted in Fig. 1. It is the classical transmission line as considered in the course. Let us for simplicity assume that there are no losses. The line is characterized by its real characteristic impedance R_c, its signal speed v and its length d along the z-axis. Note that the load plane is situated at $z = d$. In the case of Fig. 1, the generator impedance R_g and the load impedance R_L are also real. The generator $\hat{e}_g(t)$ emits, for example, a single bit. Further details about this bit are given in Appendix A.

3 Finite-Difference Time-Domain (FDTD) method

3.1 Discretization in space and time

The voltage and current along the transmission line vary continuously as a function of the space coordinate z and the time t. To numerically compute the voltage and the current, the continuous problem is discretized in space and in time:

- The first discretization is performed in space, as shown in Fig. 2. The voltage $\hat{v}(z,t)$ is computed at $N + 1$ discrete locations $z = n\Delta z$, $n = 0, \ldots, N$. The current $\hat{i}(z,t)$ is calculated at N discrete locations $z = (n + \frac{1}{2})\Delta z$, $n = 0, \ldots, N - 1$. Note that the discrete locations for the voltages and currents are staggered.

*email: dries.vandeginste@ugent.be
†email: pieter.decleer@ugent.be
‡email: dries.bosman@ugent.be

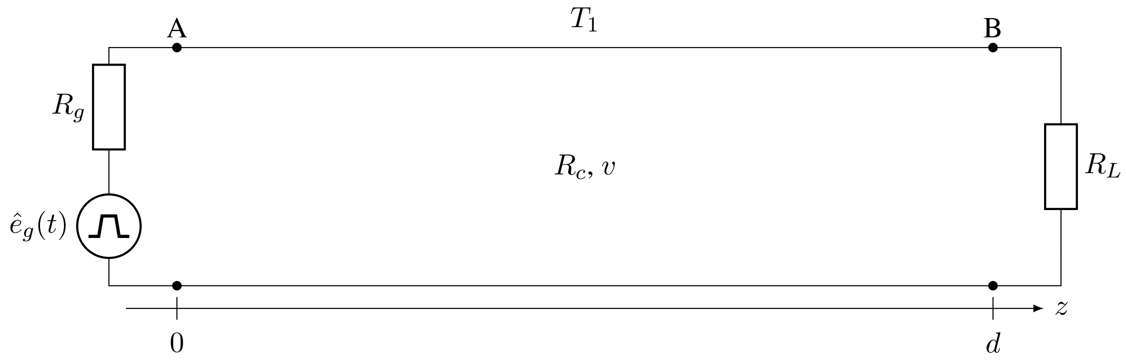

Figure 1: Problem geometry.

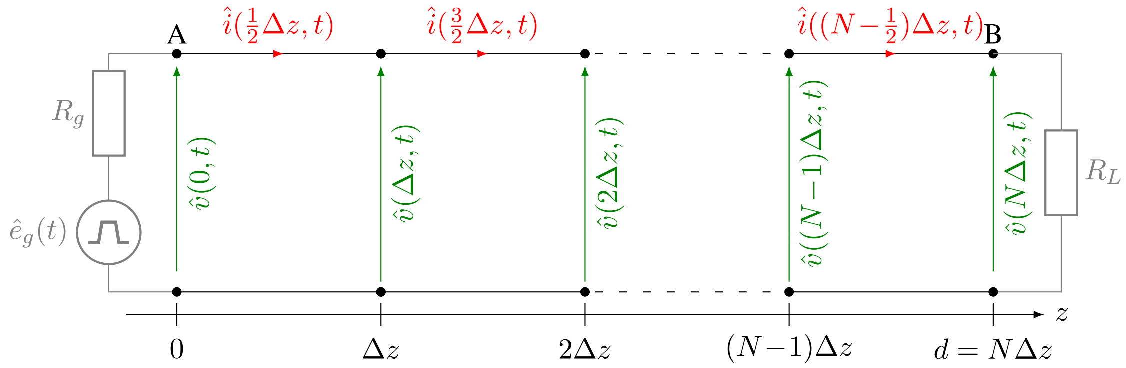

Figure 2: Discretization in space of the transmission line.

- The time axis is also discretized, and again in a staggered way. The voltages are evaluated at $M+1$ times $t = m\Delta t$, $m = 0, \dots, M$; the currents at M times $t = (m + \frac{1}{2})\Delta t$, $m = 0, \dots, M-1$.

The choice for the space step Δz and time step Δt will be discussed in Section 3.5. Now we introduce the following compact notation for the discrete voltages and currents for all n and m:

$$V_n^m \triangleq \hat{v}(n\Delta z, m\Delta t) \tag{1}$$

$$I_{n+\frac{1}{2}}^{m+\frac{1}{2}} \triangleq \hat{i}((n + \frac{1}{2})\Delta z, (m + \frac{1}{2})\Delta t). \tag{2}$$

3.2 Central finite differences

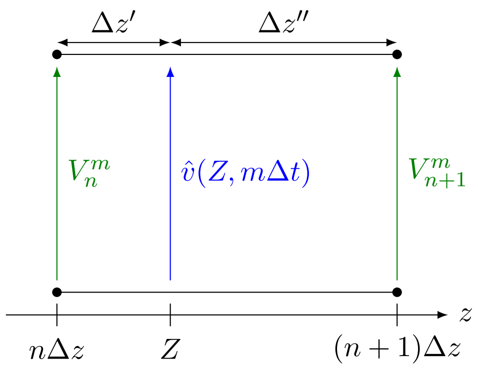

Figure 3: Relevant to the derivation of the central finite difference scheme.

We start from the telegrapher's equations for the lossless line:

$$\frac{\partial \hat{v}}{\partial z} = -L\frac{\partial \hat{i}}{\partial t}, \tag{3}$$

$$\frac{\partial \hat{i}}{\partial z} = -C\frac{\partial \hat{v}}{\partial t}, \tag{4}$$

with L and C the (known) per-unit-of-length (p.u.l.) inductance and capacitance. To derive a discretized version of the telegrapher's equations, we calculate the spatial derivative of the voltage at position Z between two voltage nodes n and $n+1$, as depicted in Fig. 3. The Taylor expansions for V_n^m and V_{n+1}^m centered about $\hat{v}(Z,t)$ are given by

$$V_n^m = \hat{v}(Z, m\Delta t) + \left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=Z} (-\Delta z') + \frac{1}{2}\left.\frac{\partial^2 \hat{v}(z, m\Delta t)}{\partial z^2}\right|_{z=Z} (-\Delta z')^2 + \mathcal{O}\left((\Delta z')^3\right), \tag{5}$$

$$V_{n+1}^m = \hat{v}(Z, m\Delta t) + \left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=Z} \Delta z'' + \frac{1}{2}\left.\frac{\partial^2 \hat{v}(z, m\Delta t)}{\partial z^2}\right|_{z=Z} (\Delta z'')^2 + \mathcal{O}\left((\Delta z'')^3\right), \tag{6}$$

where $\Delta z' = Z - n\Delta z$ and $\Delta z'' = (n+1)\Delta z - Z$. For Z chosen in the center of the nth segment, and thus $\Delta z' = \Delta z'' = \Delta z/2$, the Taylor expansions read

$$V_n^m = V_{n+\frac{1}{2}}^m + \left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=(n+\frac{1}{2})\Delta z} \frac{-\Delta z}{2} + \frac{1}{2}\left.\frac{\partial^2 \hat{v}(z, m\Delta t)}{\partial z^2}\right|_{z=(n+\frac{1}{2})\Delta z} \left(\frac{-\Delta z}{2}\right)^2 + \mathcal{O}\left((\Delta z)^3\right), \tag{7}$$

$$V_{n+1}^m = V_{n+\frac{1}{2}}^m + \left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=(n+\frac{1}{2})\Delta z} \frac{\Delta z}{2} + \frac{1}{2}\left.\frac{\partial^2 \hat{v}(z, m\Delta t)}{\partial z^2}\right|_{z=(n+\frac{1}{2})\Delta z} \left(\frac{\Delta z}{2}\right)^2 + \mathcal{O}\left((\Delta z)^3\right). \tag{8}$$

Subtracting (7) from (8) yields

$$V_{n+1}^m - V_n^m = \left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=(n+\frac{1}{2})\Delta z} \Delta z + \mathcal{O}\left((\Delta z)^3\right). \tag{9}$$

Division of this result by Δz leads to an expression for the sought-after spatial derivative, as follows:

$$\left.\frac{\partial \hat{v}(z, m\Delta t)}{\partial z}\right|_{z=(n+\frac{1}{2})\Delta z} = \frac{V_{n+1}^m - V_n^m}{\Delta z} + \mathcal{O}\left((\Delta z)^2\right). \tag{10}$$

This derivative, which is approximated by a *central* finite difference, remains *second-order* accurate. Consequently, for small Δz, we can replace the l.h.s. of (3) by (10) and maintain very good accuracy, as follows:

$$\frac{V_{n+1}^m - V_n^m}{\Delta z} = -L \left.\frac{\partial \hat{i}\left((n+\frac{1}{2})\Delta z, t\right)}{\partial t}\right|_{t=m\Delta t}. \tag{11}$$

The reader may verify that also applying a central finite difference scheme for the time derivative of the current in the r.h.s. of (11) results in

$$\frac{V_{n+1}^m - V_n^m}{\Delta z} = -L\frac{I_{n+\frac{1}{2}}^{m+\frac{1}{2}} - I_{n+\frac{1}{2}}^{m-\frac{1}{2}}}{\Delta t}. \tag{12}$$

Similarly, the second telegrapher's equation (4) can be approximated as

$$\frac{I_{n+\frac{1}{2}}^{m+1/2} - I_{n-\frac{1}{2}}^{m+1/2}}{\Delta z} = -C\frac{V_n^{m+1} - V_n^m}{\Delta t}. \tag{13}$$

From the discrete versions (12) and (13) of the telegrapher's equations, it is observed that the proposed finite difference scheme indeed leads to evaluations of voltages and currents at nodes that are staggered in space (integer n multiples of Δz for the voltage and half-integer $n+\frac{1}{2}$ multiples of Δz for the current) and in time (integer m multiples of Δt for the voltage and half-integer $m+\frac{1}{2}$ multiples of Δt for the current).

3.3 Update equations and leapfrog scheme

From (12) and (13) we derive the following so-called update equations:

$$I_{n+\frac{1}{2}}^{m+\frac{1}{2}} = I_{n+\frac{1}{2}}^{m-\frac{1}{2}} + \frac{\Delta t}{L\Delta z}\left(V_n^m - V_{n+1}^m\right), \tag{14}$$

$$V_n^{m+1} = V_n^m + \frac{\Delta t}{C\Delta z}\left(I_{n-\frac{1}{2}}^{m+\frac{1}{2}} - I_{n+\frac{1}{2}}^{m+\frac{1}{2}}\right). \tag{15}$$

These update equations allow to compute new values for the voltages and currents from the voltages and currents at previous time steps. They form the core of the so-called *leapfrog scheme*. The basic FDTD algorithm, to be implemented in this project, is thus simply as follows:

Step 1 Initialize all voltages and currents with a zero value.

Step 2 Update the currents at all N locations at time step $m + \frac{1}{2}$ using (14).

Step 3 Update the voltages at all $N + 1$ locations at time step $m + 1$ using (15).

Step 4 Iterate for all m (i.e., repeat Steps 2 and 3).

3.4 Boundary (terminal) conditions

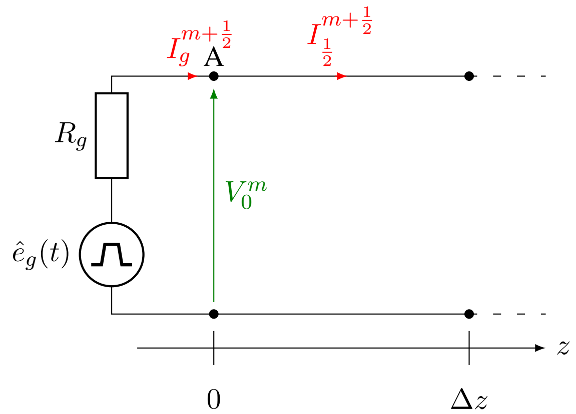

Figure 4: Relevant to the derivation of the terminal condition.

From (15) a problem arises, as the calculation of V_0^{m+1} requires the knowledge of the unexisting $I_{-\frac{1}{2}}^{m+\frac{1}{2}}$. To mitigate this issue, we replace this unexisting current by the generator current $I_g^{m+\frac{1}{2}} = \hat{\imath}(0, (m + \frac{1}{2})\Delta t)$ flowing through node A at $z = 0$ (see Fig. 4). Consequently, we also need to replace Δz by $\frac{\Delta z}{2}$ in (15)[1], yielding a new equation for the voltage at node A:

$$V_0^{m+1} = V_0^m + \frac{2\Delta t}{C\Delta z}\left(I_g^{m+\frac{1}{2}} - I_{\frac{1}{2}}^{m+\frac{1}{2}}\right). \tag{16}$$

This, however, requires the knowledge of $I_g^{m+\frac{1}{2}}$. Kirchhoff's voltage law in discretized form states that

$$I_g^{m+\frac{1}{2}} = \frac{E_g^{m+\frac{1}{2}} - V_0^{m+\frac{1}{2}}}{R_g} \tag{17}$$

$$= \frac{E_g^{m+\frac{1}{2}}}{R_g} - \frac{V_0^m + V_0^{m+1}}{2R_g}, \tag{18}$$

[1]Instead of a central difference, we performed a forward difference.

where $E_g^{m+\frac{1}{2}} = \hat{e}_g\left((m+\frac{1}{2})\Delta t\right)$ and the interpolation $V_0^{m+\frac{1}{2}} = (V_0^m + V_0^{m+1})/2$ was used as the voltages along the line are only known at integer multiples m of the time step Δt. Eliminating $I_g^{m+\frac{1}{2}}$ from (16) and (18) finally yields an update equation for V_0^{m+1}.

A similar procedure should be followed at the far-end terminal, i.e., in node B at $z = d$. You should derive both pertinent update equations for V_0^{m+1} and V_N^{m+1} yourselves, and use it in your FDTD leapfrog scheme.

3.5 Courant limit and rescaled update equations

The spatial step Δz should be small enough to resolve all wave phenomena. As a rule of thumb, one typically uses

$$\Delta z \leq \frac{\lambda_{\min}}{10}, \tag{19}$$

where $\lambda_{\min}$ is the minimal wavelength that is present in the structure. This minimal wavelength corresponds to the highest frequency that is introduced by the source. (See Appendix A for a discussion about the single bit.) For completeness it is mentioned that in two- or three-dimensional FDTD implementations, often a considerably smaller discretization (e.g., $\lambda_{\min}/30$) is chosen to avoid grid dispersion.

Once the space step is selected, an upper limit for the time step Δt is given by

$$v\Delta t \leq \Delta z. \tag{20}$$

The maximum allowed value for Δt, i.e., $\Delta z/v$, is called the *Courant limit*. It can be shown that a larger time step renders the FDTD simulations unstable and non-causal. The proof follows from an observation of the dispersion relation where the wave propagation in the discretized scheme should not exceed the speed of light. For the one-dimensional transmission line problem considered in this project, the so-called *magic time step* $v\Delta t = \Delta z$ can be used. For this time step, the discretized dispersion relation coincides with the continuous one and *no Gibbs-like phenomena are observed*.

Now we introduce the dimensionless Courant factor

$$\alpha \triangleq \frac{v\Delta t}{\Delta z}, \tag{21}$$

and we rescale the current as follows:

$$\tilde{I}_{n+\frac{1}{2}}^{m+\frac{1}{2}} = I_{n+\frac{1}{2}}^{m+\frac{1}{2}} R_c, \tag{22}$$

with $R_c = \sqrt{L/C}$. With (21) and (22), the update equations (14) and (15) can be reformulated as:

$$\tilde{I}_{n+\frac{1}{2}}^{m+\frac{1}{2}} = \tilde{I}_{n+\frac{1}{2}}^{m-\frac{1}{2}} + \alpha\left(V_n^m - V_{n+1}^m\right), \tag{23}$$

$$V_n^{m+1} = V_n^m + \alpha\left(\tilde{I}_{n-\frac{1}{2}}^{m+\frac{1}{2}} - \tilde{I}_{n+\frac{1}{2}}^{m+\frac{1}{2}}\right). \tag{24}$$

Note that these rescaled update equations only depend on one dimensionless parameter α (which should not exceed one) and that the unknowns share the same unit (Volt), making the scheme less susceptible to numerical errors.

4 Project description

At least the followings tasks should be performed:

1. Write a program in Python3 that implements the FDTD technique for the simple transmission line of Fig. 1.

2. As **input**, your code should accept a filename as a command line argument. The contents of this input file is described in Appendix B and an example can be found on Ufora (named `example.txt`). *The format of this input file should NOT be changed!!*

3. As **output**, the code should produce

   - A plot showing the voltage as a function of z at a particular time $t = T$.
   - A plot showing the voltage as a function of t at a particular position $z = Z$.

   In Appendix B it is explained how the position Z and time T are fed into the program. Examples of these plots are also found on Ufora (named `exampletime.png` and `exampleposition.png`). These plots correspond to the parameters of the example input file `example.txt`, and can thus be used to validate your own code.

4. Let the program run sufficiently long as to see what happens to the bit.

5. Repeat this for several variations of the parameters $\tau_h = T_{\text{bit}}$ and τ_r (see Appendix A), and for a few variations of the load and generator impedance. Why do the waveforms look the way they do?

6. Explore the effect of the Courant limit. For time steps that are smaller than required by the Courant limit, the results should stay identical. Do you see differences with respect to the case where the Courant limit is exactly satisfied (i.e., $v\Delta t = \Delta z$)? What happens if Δt becomes too large?

7. Now also study a load consisting of a resistive part R_L in parallel to a capacitor C_L. Thereto, adjust the pertinent update equations, adapt your code, and run the simulations. Again, why do the waveforms look the way they do?

8. Write a report that contains (at least) the following information:

   (a) brief description of the update equations for the case of a load consisting of a resistor in parallel with a capacitor.

   (b) numerical results (i.e., output of your program for several interesting choices of parameters);

   (c) thorough discussion of the obtained results and observations!!

   (d) a brief but clear indication of the contribution of each group member;

   (e) conclusions.

   The report should *not* exceed 10 pages, including figures and references.

This assignment is performed in groups of maximum three people. **Deadline 1**: Subscribe to a group of your choice via Ufora before Friday, Nov. 6, 2020 at 11.59pm. Missing the deadline means losing the free choice of group. **Deadline 2**: Upload your written report (PDF-file) + the actual Python code + the inputfiles for the simulations discussed in your report in a *single* ZIP-file named GroupX.zip, X being your group number, via "Ufora-tools: Assignments" before Friday, Dec. 18, 2020 at 11.59pm. Missing the deadline means losing marks.

Notes:

- Make sure that your code is self-contained. We will 'feed' several benchmark examples to your code, using the proper format of the input file, and check your output graphs! Also try to make the program efficient, e.g., use vector operations instead of for-loops and only store the data required for the output.

- Your report should be a self-contained PDF document.
- Self-efficacy and engineering spirit are encouraged, with extra marks for creative ideas that focus on the electromagnetic aspects of the problem. If you choose to add extra features, which require more input than what is provided in the input file, either make sure that your expanded code can still handle the basic input file `example.txt` or provide a *second* code — besides the *basic* required code — and explain to us how to use this code and the corresponding (slighty) adjusted input file.
- The project (theory + implementation + interpretation of results) can also be a topic on the exam.

Appendix A: Single bit and its spectrum

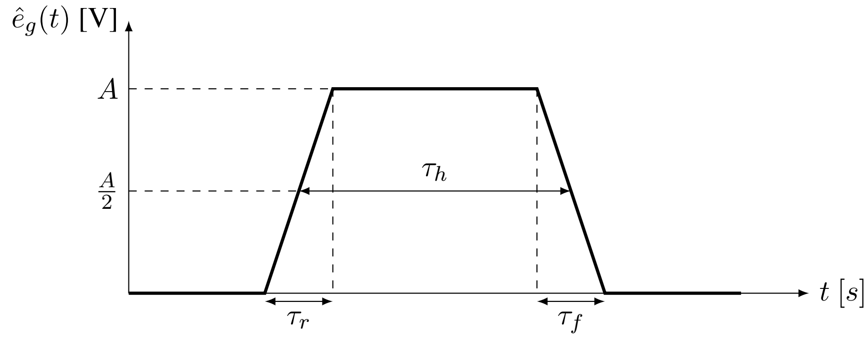

Figure 5: Voltage waveform of a single bit.

Consider the single bit's voltage waveform of Fig. 5, which is a trapezoid-shaped impulse of amplitude A and half-amplitude pulsewidth τ_h. Here, the pulse is symmetrical, as the risetime τ_r and falltime τ_f are equal. Note that a data signal (bit sequence) simply consists of a sequence of such single bits. The bit duration T_{bit} in such a sequence equals τ_h and the bitrate is given by $1/T_{\text{bit}}$.

The spectrum of a single-event pulse or transient signal is determined using the Fourier transform. For the source signal $\hat{e}_g(t)$ in the time domain, the frequency-domain counterpart is given by

$$e_g(\omega) = \int_{-\infty}^{+\infty} \hat{e}_g(t) e^{-j\omega t}\, dt, \tag{25}$$

with $\omega = 2\pi f$ the angular frequency. In the particular case of a single bit, we get

$$e_g(\omega) = A\tau_h\ \text{sinc}(\pi\tau_h f)\ \text{sinc}(\pi\tau_r f). \tag{26}$$

A piecewise linear upper bound $E_{\text{bound}}(\omega) \geq |e_g(\omega)|$ is readily derived and plotted in Fig. 6 on a log-log

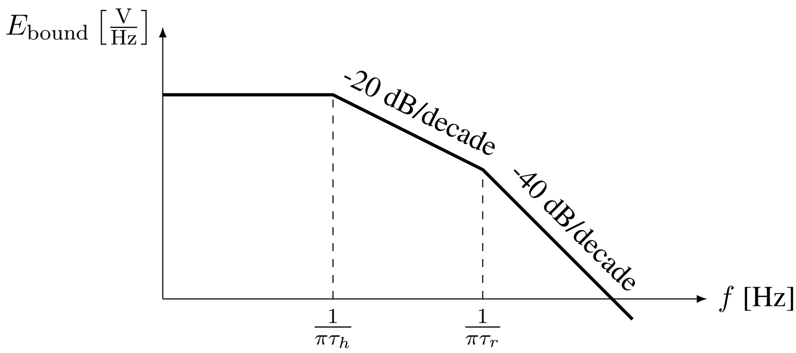

Figure 6: Log-log plot of the upper bound of the spectrum of the single bit of Fig. 5

scale. This bound indicates that the amplitude of the spectral components of the source signal does not exceed the three straight lines shown in Fig. 6. We observe two corner frequencies:

$$f_h = \frac{1}{\pi \tau_h}, \tag{27}$$

$$f_r = \frac{1}{\pi \tau_r}. \tag{28}$$

At frequencies above f_h, the asymptotic amplitude of the spectral components of the signal is inversely proportional to frequency (-20 dB/decade or -6 dB/octave). Above f_r, it is inversely proportional to the square of the frequency (-40 dB/decade or -12 dB/octave). From these spectral bounds, it now becomes clear that the high-frequency content of a single bit is due primarily to the rise-/falltime of the pulse. Therefore, the bandwidth of such a signal is often defined as

$$\mathrm{BW} = \frac{1}{\pi \tau_r}. \tag{29}$$

This bandwidth can be seen as the maximum frequency that is present in the spectrum of the signal.

Appendix B: Input file format

The input file required to run the code should be fed to the program as a command line argument. So the program is executed by typing '`python program.py inputfile`' at the command line. If you prefer to use an IDE such as spyder, you can add command line arguments by going to `Run > Configuration per file...`, then checking the box `Command line options` and typing the name of the file in the box.

An input file contains all the necessary variables to perform a simulation and can be read by using the function `numpy.loadtxt`[2]. Now, have a look at the inputfile `example.txt` as this has the structure required for your input files. Every line contains a specific variable, defined after the '#'-symbol. Most variables are self-explanatory and the only variables that could require some explanation are:

- **bit delay [s]**: This is an extra delay added to the source function, expressed in seconds.
- **position of the sensor [m]**: This is the position at which the voltage is recorded, expressed in meters. The voltage at this point is to be used for the output.
- **snapshot time**: This is the time step (integer) at which the voltage for every position is recorded. This data is to be used for the required output.

[2]Consult the documentation page.