A Finite Difference Approach to Solving the Transmission Line Telegraph Equation

Christian Y. Cahig [*]
Department of Electrical Engineering and Technology
Mindanao State University - Iligan Institute of Technology
Iligan City, Philippines
{christian.cahig}@g.msuiit.edu.ph

Abstract

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

1 Introduction

Consider a transmission line of length X characterized by per-unit-length series resistance R, series inductance L, shunt conductance G, and shunt capacitance C. Let $u(x,t)$ be the instantaneous voltage signal (referred to ground) at point x along the length of the line at time t, where $0 \leq x \leq X$ and $0 \leq t \leq T$. We refer to $x = 0$ as the *sending end* and $x = X$ as the *receiving end* of the line. From elementary transmission line theory, the propagation of a voltage signal through the line is described by the *telegraph equation*:

$$\frac{1}{LC}\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} + \left(\frac{G}{C} + \frac{R}{L}\right)\frac{\partial u(x,t)}{\partial t} + \left(\frac{RG}{LC}\right) u(x,t) \tag{1}$$

which is a hyperbolic partial differential equation (PDE) (Chapra & Canale, 2015). Letting

$$c^2 = \frac{1}{LC}, \quad \alpha = \frac{G}{C}, \quad \beta = \frac{R}{L}$$

we can rewrite Equation 1 more succinctly as

$$c^2\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} + (\alpha+\beta)\frac{\partial u(x,t)}{\partial t} + \alpha\beta u(x,t) \tag{2}$$

The first-order term on the right-hand side of Equation 2 is the *dissipation term*, while the zeroth-order term is the *dispersion term*. In the absence of losses, *i.e.*, $R = G = 0$, the telegraph equation reduces into one describing a wave that propagates at a velocity c and an angular frequency ω given as

$$\omega = \frac{1}{X\sqrt{LC}} \tag{3}$$

Solving the telegraph equation is valuable in the analysis of power system dynamics. However, deriving the expression for the exact analytic solution may not always be tractable nor the most efficient course; in which case numerical approaches that approximate the PDE as a combination of algebraic operations are used. This work presents a basic finite difference method for numerically solving the transmission line telegraph equation.

The remainder of the paper proceeds as follows. Section 2 details how the telegraph equation is approximated as a linear equation via discretization and finite differences. Section 3 presents and discusses results from select worked examples. Section 4 concludes the work.

[*]Under the supervision of Engr. Michael S. Villame.

2 Finite Difference Approximation

2.1 Discretization Scheme

We transform the continuous spatial domain into a set of equally separated discrete points, *i.e.*,

$$0 \le x \le X \quad \longrightarrow \quad x_k = k\Delta x,\ \ 0 \le k \le K \in \mathbb{Z}$$

In other words, we approximate the spatial domain by sampling $K+1$ points spaced Δx apart. Note that x_0 corresponds to $x = 0$ just as x_K to $x = X$. Similarly, for the temporal domain:

$$0 \le t \le T \quad \longrightarrow \quad t_n = n\Delta t,\ \ 0 \le n \le N \in \mathbb{Z}$$

where t_0 corresponds to $t = 0$ as t_N to $t = T$. The voltage defined on the continuous domain is likewise discretized, and is parametrized by k and n:

$$u\left(x,t\right) \quad \longrightarrow \quad u\left(x_k,t_n\right)$$

For notational convenience, $u_k^n = u\left(x_k,t_n\right)$.

2.2 Difference Equation

We can approximate the continuous derivatives as central divided differences:

$$\begin{aligned}
\frac{\partial u\left(x,t\right)}{\partial t} &\longrightarrow & \frac{\partial u_k^n}{\partial t} &= \frac{u_k^{n+1}-u_k^{n-1}}{2\Delta t}\\
\frac{\partial^2 u\left(x,t\right)}{\partial t^2} &\longrightarrow & \frac{\partial 2u_k^n}{\partial t^2} &= \frac{u_k^{n+1}-2u_k^{n}+u_k^{n-1}}{\left(\Delta t\right)^2}\\
\frac{\partial^2 u\left(x,t\right)}{\partial x^2} &\longrightarrow & \frac{\partial^2 u_k^n}{\partial x^2} &= \frac{u_{k+1}^{n}-2u_k^{n}+u_{k-1}^{n}}{\left(\Delta x\right)^2}
\end{aligned}$$

Substituting these into their continuous counterparts, we approximate the telegraph as a difference equation:

$$c^2\frac{u_{k+1}^{n}-2u_k^{n}+u_{k-1}^{n}}{\left(\Delta x\right)^2} = \frac{u_k^{n+1}-2u_k^{n}+u_k^{n-1}}{\left(\Delta t\right)^2} + \left(\alpha+\beta\right)\frac{u_k^{n+1}-u_k^{n-1}}{2\Delta t} + \alpha\beta u_k^n \tag{4}$$

This numerical approximation of Equation 2 suggests that we can estimate the voltage at point x_k at the next time instant t_{n+1} given the voltages at x_k and at the neighbouring points at the current time instant (that is, u_k^n, u_{k-1}^n, and u_{k+1}^n) and the voltage at x_k at the preceding time instant (that is, u_k^{n-1}).

2.3 Update Scheme

From Equation 4, we can obtain the “update” u_k^{n+1} given u_k^n, u_{k-1}^n, u_{k+1}^n, and u_k^{n-1}. To express this more explicity, we can rewrite Equation 4 as

$$Au_k^{n+1} = Eu_{k-1}^n + Fu_k^n + Eu_{k+1}^n - Bu_k^{n-1} \tag{5}$$

where

$$A = \ 1 + \frac{\Delta\left(\alpha+\beta\right)}{2} \tag{6}$$

$$B = \ 1 - \frac{\Delta\left(\alpha+\beta\right)}{2} \tag{7}$$

$$E = \left(c\frac{\Delta t}{\Delta x}\right)^2 \tag{8}$$

$$F = 2 - 2\left(c\frac{\Delta t}{\Delta x}\right)^2 - \alpha\beta\left(\Delta t\right)^2 \tag{9}$$

2.4 Some Remarks

2.4.1 Encoding Initial and Boundary Conditions

Notice that the difference equation approximation applies for $k = 1, 2, \ldots, K-1$ and $n = 1, 2, \ldots, N-1$. It requires initial (*i.e.*, at t_0) and boundary (*i.e.*, at x_0 and x_K) values to be specified separately.

In general, initial voltage values are exressed as a function of x:

$$u(x,0) = \mu(x) \quad \longrightarrow \quad u_k^0 = \mu(x_k), \ \forall k.$$

It is also common to have predetermined initial time rate of change of voltage, which can then be approximated by a forward finite divided difference:

$$\frac{\partial u(x,0)}{\partial t} = \xi^0 \quad \longrightarrow \quad \frac{\partial u_k^0}{\partial t} = \frac{u_k^1 - u_k^0}{\Delta t} = \xi^0 \quad \longrightarrow \quad u_k^1 = u_k^0 + \xi^0 \Delta t, \ \forall k.$$

The sending- and receiving-end voltages can be expressed as functions of t:

$$\begin{aligned} u(0,t) &= \nu_0(t) \quad &\longrightarrow \quad u_0^n &= \nu_0(t_n), \ \forall n \\ u(X,t) &= \nu_X(t) \quad &\longrightarrow \quad u_K^n &= \nu_X(t_n), \ \forall n. \end{aligned}$$

Information at the boundaries may also be expressed in terms of space-derivatives, which can be approximated usign forward and backward finite divided differences:

$$\begin{aligned} \frac{\partial u(0,t)}{\partial x} &= \gamma_0 \quad \longrightarrow \quad \frac{\partial u_0^n}{\partial x} = \frac{u_1^n - u_0^n}{\Delta x} = \gamma_0 \quad &\longrightarrow \quad u_0^n &= u_1^n - \gamma_0 \Delta x, \ \forall n \\ \frac{\partial u(X,t)}{\partial x} &= \gamma_X \quad \longrightarrow \quad \frac{\partial u_K^n}{\partial x} = \frac{u_K^n - u_{K-1}^n}{\Delta x} = \gamma_X \quad &\longrightarrow \quad u_K^n &= u_{K-1}^n + \gamma_X \Delta x, \ \forall n. \end{aligned}$$

2.4.2 Vectorizing the Update Scheme

The update scheme Equation 5 is essentially a system of $K-1$ linear equations:

$$A \begin{bmatrix} u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{K-1}^{n+1} \end{bmatrix} = \begin{bmatrix} E & F & E & 0 & \cdots & 0 & 0 & 0 \\ 0 & E & F & E & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & E & F & E \end{bmatrix} \begin{bmatrix} u_0^n \\ u_1^n \\ u_2^n \\ \vdots \\ u_{K-1}^n \\ u_K^n \end{bmatrix} - B \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} u_0^{n-1} \\ u_1^{n-1} \\ u_2^{n-1} \\ \vdots \\ u_{K-1}^{n-1} \\ u_K^{n-1} \end{bmatrix}$$

Letting

$$\begin{aligned} \tilde{\mathbf{u}}^n &= \left[u_1^n, u_2^n, \ldots, u_{K-1}^n\right]^\mathsf{T} \in \mathbb{R}^{K-1} \\ \mathbf{u}^n &= \left[u_0^n, u_1^n, \ldots, u_{K-1}^n, u_K^n\right]^\mathsf{T} \in \mathbb{R}^{K+1} \\ \mathbf{E} &= \begin{bmatrix} E & F & E & 0 & \cdots & 0 & 0 & 0 \\ 0 & E & F & E & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & E & F & E \end{bmatrix} \in \mathbb{R}^{(K-1)\times(K+1)} \qquad (10) \\ \mathbf{B} &= B\left[\mathbf{0} \quad \mathbf{I} \quad \mathbf{0}\right] \in \mathbb{R}^{(K-1)\times(K+1)} \qquad (11) \end{aligned}$$

where $\mathbf{0}$ is an $(K-1)$-vector of zeros and $\mathbf{I}$ is the identity matrix of size $(K-1)$, the update equations can be expressed compactly as

$$A\tilde{\mathbf{u}}^{n+1} = \mathbf{E}\mathbf{u}^n - \mathbf{B}\mathbf{u}^{n-1} \qquad (12)$$

$$\mathbf{u}^{n+1} = \begin{bmatrix} u_0^{n+1} \\ \tilde{\mathbf{u}}^{n+1} \\ u_K^{n+1} \end{bmatrix} \qquad (13)$$

where u_0^{n+1} and u_K^{n+1} are determined from the boundary conditions.

2.4.3 Courant-Friedrichs-Levy (CFL) Condition

In general, the finer the spatial and temporal domains are discretized, the better u_k^n approximates $u(x,t)$. However, as Δx and Δt get smaller, the number of gridpoints at which $u(x,t)$ is to be approximated increases, and so does the computational burden. Moreover, a judicious choice of the spatial and temporal steps helps avoid instability (*i.e.*, when the error drastically accumulates). The CFL condition is a commonly used guide for selecting Δx or Δt (given the other):

$$\epsilon = c\frac{\Delta t}{\Delta x} \leq 1 \tag{14}$$

where ϵ is called the *CFL number*. In other words, for a particular Δx, the time step should be

$$\Delta t \leq \frac{\Delta x}{c}$$

to avoid an unstable apporximation. Intuitively, this upper limit on Δt says that the simulation cannot be incremented any more than the time required for a wave to travel one grid step in space.

3 Illustrative Examples

To supplement the discussion in the preceding sections, we present a few examples that focus on different aspects of the finite difference numerical approximation of the transmission line telegraph equation. We first investigate the benefits of using the vectorized update scheme (Equation 10–13) in Section 3.1. Then, in Sections 3.2 and 3.3, we look into the effects of varying how "fine" the spatial or the temporal domains are discretized. Lastly, we simulate a bus fault event and how it is cleared in Sections 3.4 and 3.5, respectively. All procedures are accomplished using NumPy (Harris et al., 2020) on an Intel® Core™ i7-10750H with 16GB of RAM. Source codes and related files are available in `illustrative examples/` within the project repository.

3.1 On Vectorization

Consider the scenario modelled as

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} + 3.00001\frac{\partial u(x,t)}{\partial t} + 3\times 10^{-5} u(x,t) \tag{15}$$

where $0 \leq x \leq 1$, $0 \leq t \leq 1$, and subject to

$$u(x,0) = \sin(5\pi x) + 2\sin(7\pi x)\,,\ \forall x \tag{16}$$

$$\frac{\partial u(x,0)}{\partial t} = 0,\ \forall x \tag{17}$$

$$u(0,t) = 0,\ \forall t \tag{18}$$

$$u(1,t) = 0,\ \forall t \tag{19}$$

Ten x-domain discretizations are examined; we start from $K+1 = 100$ grid points and increment by 50. To avoid numerical instability, we use $\epsilon = 0.10$ to set the corresponding time steps. For each domain discretization, the telegraph equation is solved using both the basic (Equation 5–9) and the vectorized (Equation 10–13) update schemes. Twenty independent runs are performed, where execution times are the metrics of interest. All procedures are documented in `illustrative examples/on vectorization/on vectorization.ipynb`.

Runtimes for the different discretizations and update schemes are summarized in Figure 1. The vectorized update scheme is significantly faster than the naive loop approach, and the speedup gained increases as the domain discretization gets finer. The disparity in the execution times is attributed to NumPy's support for array-based computing; in fact, one can expect to arrive at similar findings when using other numerical computing tools such as MATLAB® (The MathWorks Inc., 2020). This suggests that, for the same computational resources and time duration, vectorization enables one to iterate and model more scenarios than using the explicit loop-based approach. Henceforth, we will use the vectorized update scheme.

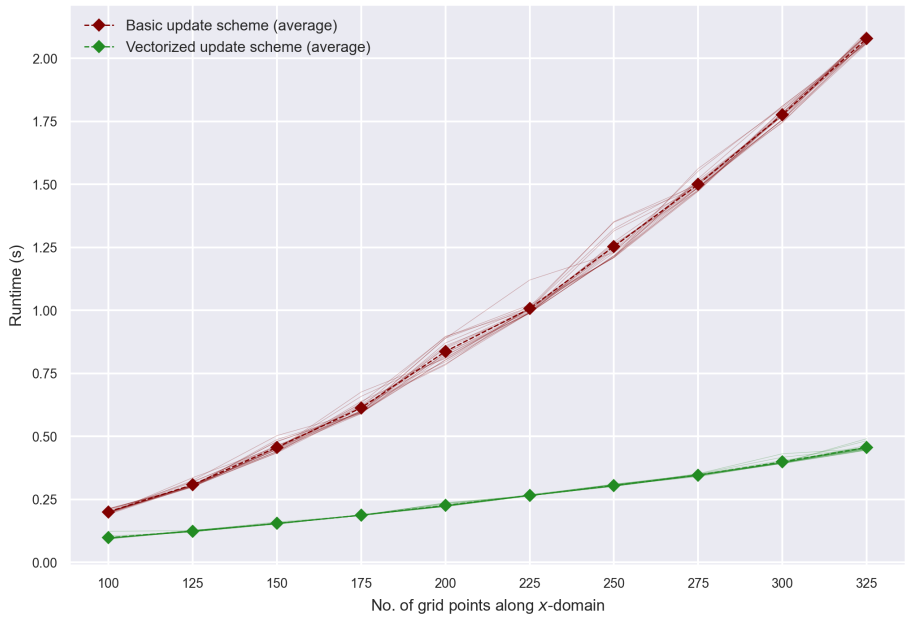

Figure 1: Execution times for basic and vectorized update schemes at different domain discretizations.

3.2 On Spatial Domain Discretization

Consider the scenario

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial^2 u(x,t)}{\partial t^2} + 3\frac{\partial u(x,t)}{\partial t} \tag{20}$$

$$u(x,0) = \sin(5\pi x) + 2\sin(7\pi x),\ \forall x \tag{21}$$

$$\frac{\partial u(x,0)}{\partial t} = 0,\ \forall x \tag{22}$$

$$u(0,t) = 0,\ \forall t \tag{23}$$

$$u(1,t) = 0,\ \forall t \tag{24}$$

where $0 \le x \le 1$ and $0 \le t \le 1$. From Zhang et al. (2019), the corresponding analytic solution is

$$\begin{aligned} u(x,t) = e^{-1.5t}\sin(5\pi x)\left[\cos(\theta(5)t) + \frac{1.5}{\theta(5)}\sin(\theta(5)t)\right] \\ + 2e^{-1.5t}\sin(7\pi x)\left[\cos(\theta(7)t) + \frac{1.5}{\theta(7)}\sin(\theta(7)t)\right] \end{aligned} \tag{25}$$

$$\theta(z) = \sqrt{(z\pi)^2 - 1.5^2} \tag{26}$$

To illustrate how discretization of the spatial domain affects the quality of the numerical approximation, we consider varying K (and hence, the number of grid points) while fixing $N = 999$ (that is, 1000 grid points along the temporal domain). We compare the results obtained using the finite difference approximations and those obtained by the analyic solution (Equation 25–26); in particular, we look at the mean squared error (MSE) calculated as

$$\sqrt{\sum_{n=0}^{N}\sum_{k=0}^{K}\{u(x_k,t_n) - \hat{u}(x_k,t_n)\}^2}$$

where $u(x_k,t_n)$ and $\hat{u}(x_k,t_n)$ are the respective analytic and approximate instantaneous voltages evaluated at the grid points along $x-$ and $t-$domains. All procedures are documented in `illustrative examples/on spatial domain discretization/on spatial domain discretization.ipynb`.

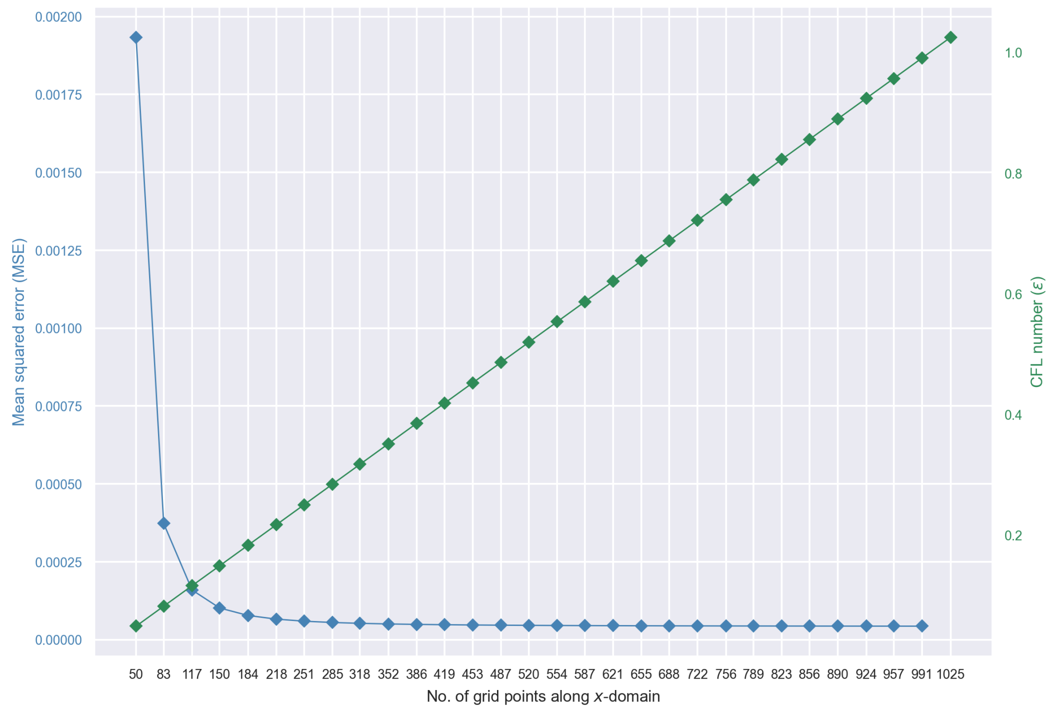

Figure 2: MSE and ϵ for various discretization schemes of the x-domain.

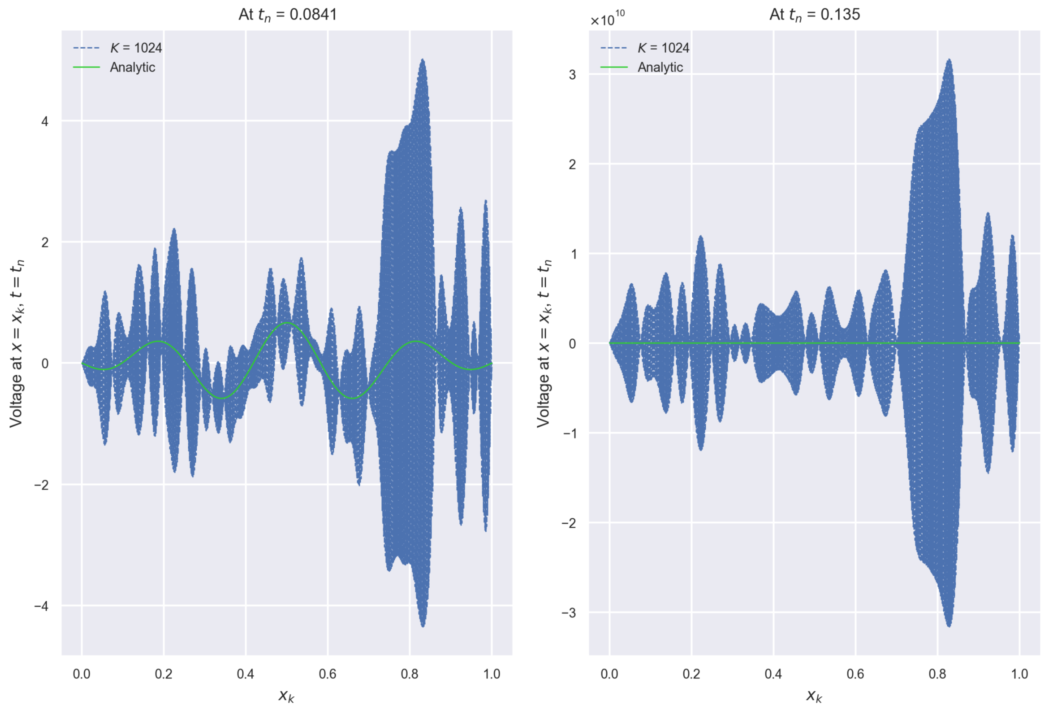

Figure 3: Numerical instability when the $x-$domain is approximated by 1025 grid points and the $t-$domain by 1000 grid points.

Referring to Figure 2, we find a general trend of the approximation accuracy improving as more grid points are sampled from the spatial domain. One can explain this by considering the limit definition of the partial

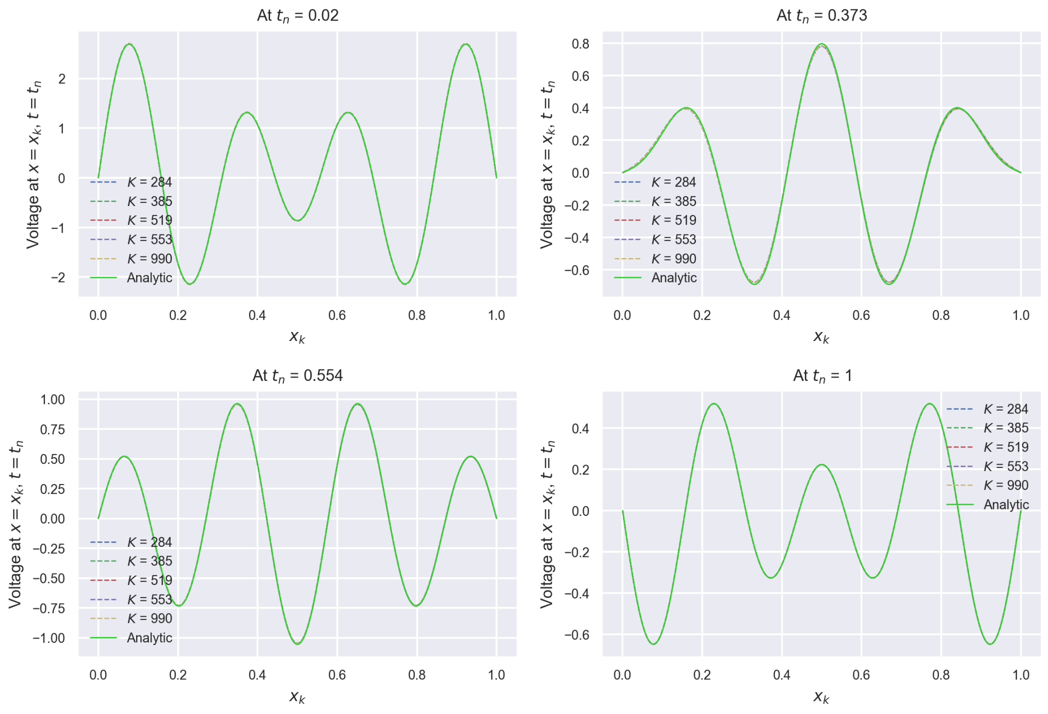

Figure 4: Comparing analytic solution with finite difference approximations at various K.

derivative with respect to x:

$$\frac{\partial u(x,t)}{\partial x} = \lim_{\Delta x \longrightarrow 0} \frac{u(x + \Delta x, t)}{\Delta x} \tag{27}$$

Notice also the diminishing returns in accuracy with increasing K: there is a significant drop in MSE as one increases the number of grid points from 50 to 150, but the improvement tapers off and is insignificant from about 450 to 990 grid points. From a practical viewpoint, one can specify a "good enough" accuracy level so that the number of grid points–and hence, the computational expenses–can be kept to a manageable value. Another salient observation from Figure 2 is that the MSE increases without bounds when 1025 grid points are sampled from the $x-$domain. This "finite time blowup" (see Figure 3) is expected to occur since

$$\epsilon = c\frac{\Delta t}{\Delta x} = \frac{\frac{1-0}{1000-1}}{\frac{1-0}{1025-1}} = \frac{1024}{999} > 1$$

Nevertheless, the finite difference approximation performs well when the number of spatial grid points yields a numerically stable solution (see Figure 4).

3.3 On Temporal Domain Discretization

Consider again the scenario in Equation 20–24. This time we illustrate how the discretization of the temporal domain affects the quality of the numerical approximation. Specifically, we vary N while fixing $K = 99$ (that is, 100 grid points along the spatial domain). We compare the results obtained using the finite difference approximations and those obtained by the analytic solution (Equation 25–26) in terms of the MSE. All procedures are documented in `illustrative examples/on temporal domain discretization/on temporal domain discretization.ipynb`.

We see in Figure 5 that, for a fixed discretization of the spatial domain, the finite difference approximation generally improves (*i.e.*, the MSE decreases) as more grid points are sampled from the temporal domain. This can be explained by considering the limit definition of the partial time derivative:

$$\frac{\partial u(x,t)}{\partial t} = \lim_{\Delta t \longrightarrow 0} \frac{u(x, t + \Delta t)}{\Delta t} \tag{28}$$

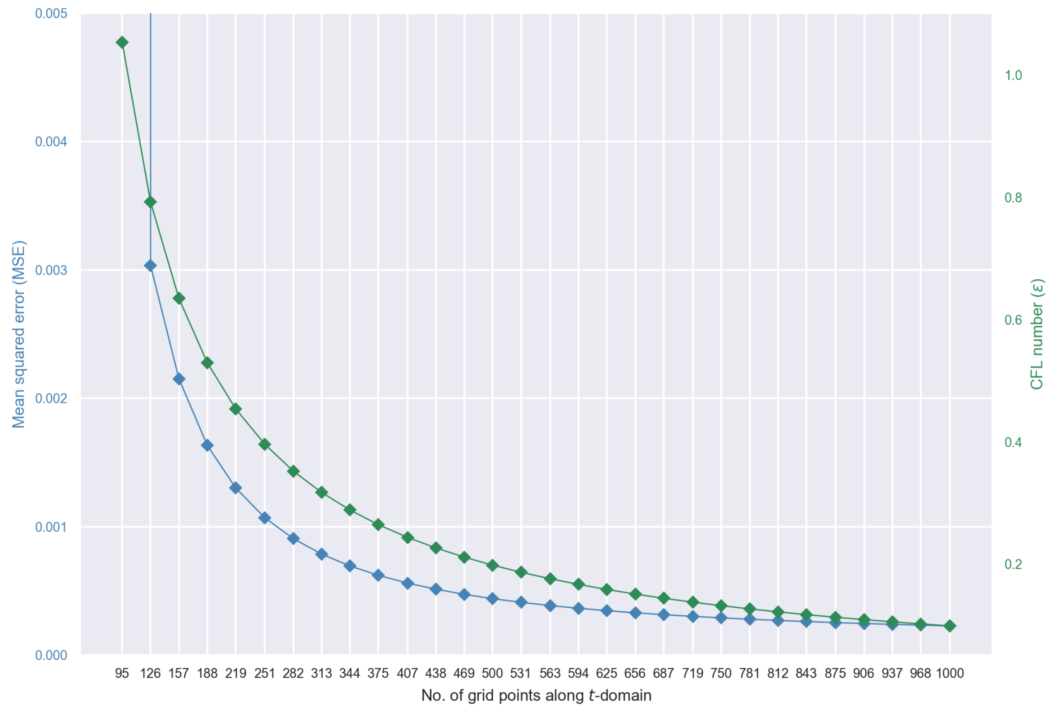

Figure 5: MSE and ϵ for various discretization schemes of the t-domain.

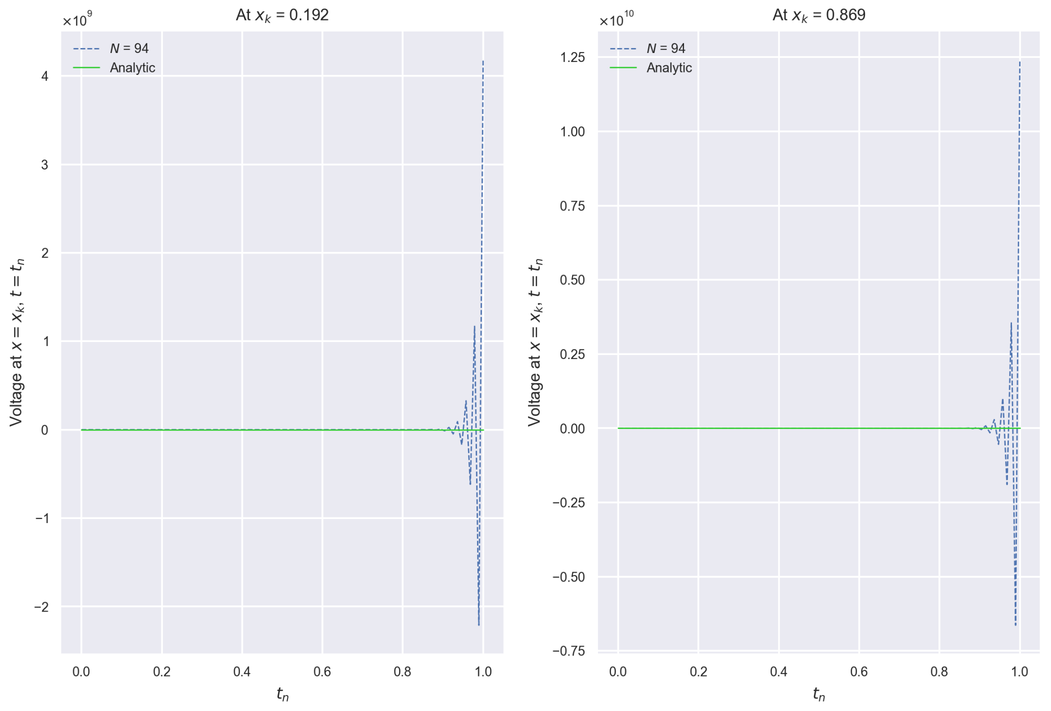

Figure 6: Numerical instability when the $x-$domain is approximated by 100 grid points and the $t-$domain by 95 grid points.

However, we note that even if we decrease Δt to zero (*i.e.*, the continuous limit) the interpolation error is still nonzero; hence, the MSE never quite reaching null. Similar to the case with $x-$domain discretization,

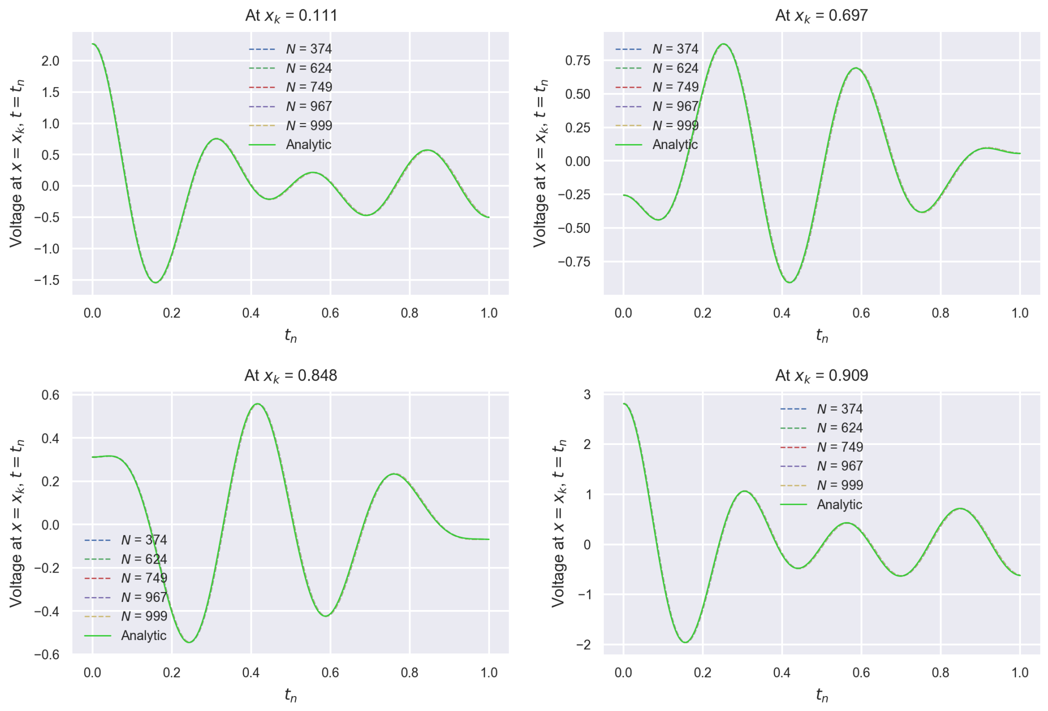

Figure 7: Comparing analytic solution with finite difference approximations at various N.

the accuracy gain tapers off with increasing N. From a practical perspective, one can attain approximately the same MSE when using 750 and 1000 grid points, with the former being computationally lighter. We also observe a sudden increase in MSE when 95 grid points are sampled from the $t-$domain. This error accumulation, better illustrated in Figure 6, is an expected outcome since

$$\epsilon = c\frac{\Delta t}{\Delta x} = \frac{\frac{1-0}{95-0}}{\frac{1-0}{100-0}} = \frac{100}{95} > 1$$

When the number of temporal grid points yields $\epsilon <= 1$, the finite difference approximation performs well (see Figure 7).

3.4 Simulating a Faulted Bus

asd

Phasellus vestibulum orci vel mauris. Fusce quam leo, adipiscing ac, pulvinar eget, molestie sit amet, erat. Sed diam. Suspendisse eros leo, tempus eget, dapibus sit amet, tempus eu, arcu. Vestibulum wisi metus, dapibus vel, luctus sit amet, condimentum quis, leo. Suspendisse molestie. Duis in ante. Ut sodales sem sit amet mauris. Suspendisse ornare pretium orci. Fusce tristique enim eget mi. Vestibulum eros elit, gravida ac, pharetra sed, lobortis in, massa. Proin at dolor. Duis accumsan accumsan pede. Nullam blandit elit in magna lacinia hendrerit. Ut nonummy luctus eros. Fusce eget tortor.

3.5 Simulating a Fault-Clearing Scenario

Ut sit amet magna. Cras a ligula eu urna dignissim viverra. Nullam tempor leo porta ipsum. Praesent purus. Nullam consequat. Mauris dictum sagittis dui. Vestibulum sollicitudin consectetuer wisi. In sit amet diam. Nullam malesuada pharetra risus. Proin lacus arcu, eleifend sed, vehicula at, congue sit amet, sem. Sed sagittis pede a nisl. Sed tincidunt odio a pede. Sed dui. Nam eu enim. Aliquam sagittis lacus eget libero. Pellentesque diam sem, sagittis molestie, tristique et, fermentum ornare, nibh. Nulla et tellus non felis imperdiet mattis. Aliquam erat volutpat.

4 Conclusion

Nulla ac nisl. Nullam urna nulla, ullamcorper in, interdum sit amet, gravida ut, risus. Aenean ac enim. In luctus. Phasellus eu quam vitae turpis viverra pellentesque. Duis feugiat felis ut enim. Phasellus pharetra, sem id porttitor sodales, magna nunc aliquet nibh, nec blandit nisl mauris at pede. Suspendisse risus risus, lobortis eget, semper at, imperdiet sit amet, quam. Quisque scelerisque dapibus nibh. Nam enim. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nunc ut metus. Ut metus justo, auctor at, ultrices eu, sagittis ut, purus. Aliquam aliquam.

References

Steven C. Chapra and Raymond P. Canale. *Numerical Methods for Engineers*. McGraw-Hill Education, seventh edition, 2015.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825): 357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

The MathWorks Inc. *MATLAB R2020b (version 9.9)*. Natick, Massachusetts, 2020.

Bolong Zhang, Wenjian Yu, and Michael Mascagni. Revisiting Kac's Method: A Monte Carlo Algorithm for Solving the Telegrapher's Equations. *Mathematics and Computers in Simulation*, 156:178–193, February 2019. doi: 10.1016/j.matcom.2018.08.007.