## References

1. L. Sevgi, "Innumeracy: The Meaning of the Numbers We Use," *IEEE Antennas and Propagation Magazine*, **49**, 2, April 2007, pp. 195-190.

2. L. B. Felsen and A. H. Kamel, "Hybrid Ray-Mode Formulation of Parallel Plate Waveguide Green's Functions," *IEEE Transactions on Antennas and Propagation*, **AP-29**, 4, July 1981, pp. 637-649.

3. L. B. Felsen, F. Akleman, L. Sevgi, "Wave Propagation Inside a Two-dimensional Perfectly Conducting Parallel Plate Waveguide: Hybrid Ray-Mode Techniques and Their Visualizations," *IEEE Antennas and Propagation Magazine*, **46**, 6, December 2004, pp. 69-89.


# A Review of Discrete Solutions of Poisson, Laplace, and Wave Equations


***Levent Sevgi***

Doğuş University, Electronics and Communication Engineering Department
Zeamet Sokak, No 21, Acıbadem – Kadıköy, Istanbul, Turkey
E-mail: lsevgi@dogus.edu.tr, levent.sevgi@ieee.org



## Abstract

One- and two-dimensional discrete solutions of Poisson, Laplace, and wave equations are given in this tutorial. The terms wave propagation and numerical propagation are discussed. Simple *MATLAB* scripts are also supplied.



Keywords: Poisson equations; Laplace equations; propagation; discretization; numerical analysis; FDTD methods; iterative methods; Taylor expansion; difference equations


## 1. Introduction

The *wave equation* is a second-order linear partial differential equation that describes the propagation of a variety of electromagnetic, acoustic, and fluid *waves*. The *Poisson/Laplace equation*, on the other hand, is a partial differential equation that describes the behavior of electric, gravitational, and fluid *potentials*. They describe some phenomena in electrodynamics and electrostatics, respectively. What are the fundamental differences between wave and Poisson/Laplace equations? What do they represent physically or numerically? What type of boundary conditions do they require? Both of them can be put into discrete and iterative forms, and can be solved numerically. People who simulate both the wave equation and Laplace equations use the term numerical propagation. To what does the term numerical propagation refer when waves or potentials are of interest? What do we mean when we say "wave propagation," "numerical wave propagation," or "numerical propagation?" This tutorial simply covers the answers to these questions, together with a few interesting *MATLAB* scripts and examples.

## 2. Taylor's Expansion and Finite Difference Discretization of the Differential Operator

Any continuous, infinitely differentiable function $f(x)$ can be represented as an infinite sum of terms, calculated from its derivatives at a single point, $x_0$, as [1]

$$f(x)=f(x_0)+\frac{(x-x_0)}{1!}\frac{df(x_0)}{dx}+\frac{(x-x_0)^2}{2!}\frac{d^2f(x_0)}{dx^2}+\frac{(x-x_0)^3}{3!}\frac{d^3f(x_0)}{dx^3}+....+\frac{(x-x_0)^n}{n!}\frac{d^nf(x_0)}{dx^n}+.... \quad (1)$$

This sum is called a Taylor expansion, and is used to replace the mathematical derivative with a finite-difference approximation. If $x_0=0$, then the sum is called the MacLaurin series. A Taylor expansion is used to approximate a function with a given accuracy. If $x$ is close to $x_0$, then a few terms would be adequate within a specified accuracy (e.g., less than a relative error of 1%-2%). More and more terms would be required for the same specified accuracy as $x$ goes further and further away from $x_0$.

A Taylor expansion is also used in the numerical solution of ordinary differential equations (ODE). For example, analytical exact and four-term numerical solutions of the ordinary differential equation

$$dy/dx=-2x-y$$

with $y(0)=-1$ are

$$y(x)=-3e^{-x}-2x+2$$

and

$$y(x)=-1+\Delta x-3\Delta x^2/2+\Delta x^3/2.$$

The numerical solution that is obtained by using $y(0)$, $y'(0)$, $y''(0)$, $y'''(0)$, etc., gives

$$y_N(0.1)=-0.915,$$

$$y_N(0.2)=-0.864,$$

$$y_N(0.5)=-0.815,$$

while the analytical solutions are

$$y_A(0.1)=-0.915,$$

$$y_A(0.2)=-0.856,$$

$$y_A(0.5)=-0.810.$$

The mathematical definition of a derivative is given as

$$f'(x_0)=\frac{df}{dx}=\lim_{\Delta x\to 0}\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}. \quad (2)$$

Numerically, Equation (2) can be replaced with

$$f'(x_0)\cong\frac{f(x_0+\Delta x)-f(x_0)}{\Delta x}+O(\Delta x), \quad (3)$$

which may easily be obtained from Equation (1) if $\Delta x=x-x_0$. Here, $O(\Delta x)$ refers to the order of the error (the most significant term of the residue). This is called the *forward-difference* scheme, because only points $x_0$ and $x+x_0$ are used. Similarly,

$$f'(x_0)\cong\frac{f(x_0)-f(x_0-\Delta x)}{\Delta x}+O(\Delta x), \quad (4)$$

$$f'(x_0)=\frac{f(x_0+\Delta x)-f(x_0-\Delta x)}{2\Delta x}+O(\Delta x^2), \quad (5)$$

are the *backward-* and *central-difference* schemes, respectively. The central-difference scheme uses one point more, but the error is one order less than for the forward- or backward-difference schemes. The order of the error may be further decreased if the number of points is increased.

Similarly, finite-difference representations of second derivatives are

$$f''(x_0)=\frac{f(x_0+2\Delta x)-2f(x_0-\Delta x)+f(x_0)}{\Delta x^2}, \quad (6)$$

$$f''(x_0)=\frac{f(x_0+\Delta x)-2f(x_0)+f(x_0-\Delta x)}{\Delta x^2}, \quad (7)$$

for the forward- and central-difference approximations, yielding errors of the orders of $O(\Delta x)$ and $O(\Delta x^2)$, respectively.

## 3. Poisson/Laplace/Wave Equations

The elliptic-type Poisson equation is a partial differential equation widely used to represent physical problems in electrical and mechanical engineering, theoretical physics, etc. It represents various physical problems: the temperature distribution in thermodynamics, gravitational fields and a vibrating string in mechanics, the potential distribution in electrostatics, etc. It is given as

$$\nabla^2U(x,y,z)=f(x,y,z), \quad (8)$$

where, in three-dimensional (3D) Cartesian coordinates, the Laplacian operator, $\nabla^2$ (or, as is widely used, $\Delta$) is given as

$$\nabla^2=\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}. \quad (9)$$

If the right-hand side of Equation (8) is the Dirac delta function, $\delta(x-x')(y-y')(z-z')$, then the Green's function, $g(x,y,z;x',y',z')$, replaces the function $U$. The Poisson equation is not a basic equation, but it follows directly from Maxwell's equations if all time derivatives are zero, i.e., for electrostatic conditions. In electrostatics, the Poisson equation expresses Gauss's Law: $U$ is the scalar potential function, and is obtained from the solution of Equation (8) under a given charge density distribution function, $f$ (i.e., $f=-\rho/\varepsilon$, where $\rho$ and $\varepsilon$ are the volume charge density and permittivity of the region, respectively). On the

other hand, in elasticity, $U$ in one dimension denotes the transverse displacement along the direction of a stretched string under a nonzero tension, $T$, with the right-hand side of Equation (8) being $f = w/T$, and with $w$ being the force density. Equation (8) becomes the Laplace equation for vanishing $f$.

The wave equation,

$$\left(\nabla^2 - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right)U(x,y,z;t) = 0, \tag{10}$$

represents waves (spherical in three dimensions, cylindrical in two dimensions) in the time domain in electromagnetics, and $c = 1/\sqrt{\varepsilon_0 \mu_0}$ is the velocity of light in free space. Note that for a specified geometry, Equation (8) represents a boundary-value problem (BVP), but Equation (10) is an initial-boundary value problem (IBVP).

## 4. Iterative Solutions and *MATLAB*-Based Simulators

The Poisson, Laplace, and wave equations can be solved numerically. One way is to discretize and put them into iterative forms.

### 4.1 Poisson/Laplace Equation

Suppose the Poisson equation is considered in one dimension along the $z$ direction in a finite region, $0 \le z \le Z_m$. The differential equation then reduces to

$$\frac{d^2}{dz^2}U(z) = f(z), \tag{11}$$

and can be uniquely solved if the boundary conditions, $U(0)$ and $U(Z_m)$, are supplied. The geometry of the problem is given in Figure 1. The finite-difference solution of Equation (11) is

$$U(k) = \frac{1}{2}\left[U(k+1) + U(k-1) - f(k)\right], \tag{12}$$

and it is applied to all inner nodes (i.e., $k = 2,3,4,...,KE-1,KE$ being the number of nodes). This states that the potential value at a node is determined by the values at two neighboring nodes and a given charge distribution. The terminating values, $U(1)$ and $U(KE)$, should be supplied as boundary conditions. In two dimensions, Equations (11) and (12) become

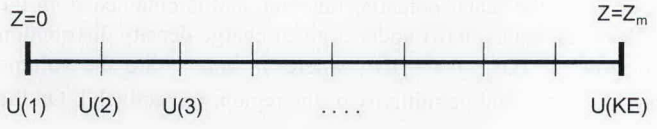


**Figure 1. A one-dimensional, finite, $z$ space.**

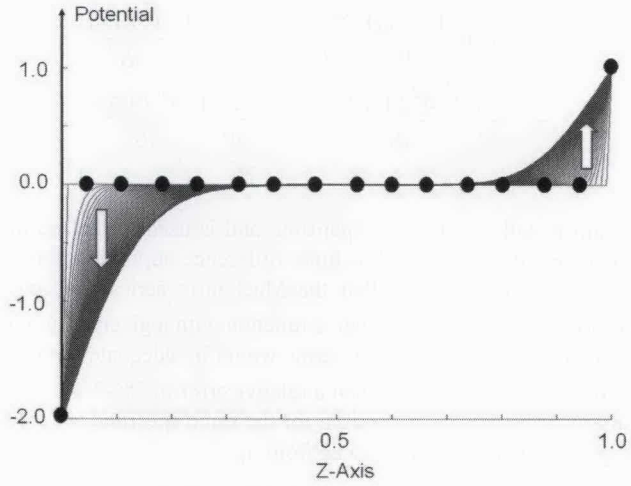


**Figure 2. A numerical solution of the Laplace equation for $Z_m = 1$, $KE = 100$, $U(0) = -2$, $U(1) = 1$, $\gamma = 62$, $\gamma \le 10^{-3}$.**

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)U(x,y) = f(x,y), \tag{13}$$

$$U_{i,j} = \frac{1}{4}\left[U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1} - f_{i,j}\right]. \tag{14}$$

This states that the potential at a node can be calculated from the potentials at four neighboring nodes plus a given charge distribution. Figure 2 shows the results of a Laplace problem with $Z_m = 1$, $U(0) = -2$, and $U(1) = 1$. This means that the potentials at the left and right are –2V and 1V, respectively. The problem is to find the $U(x)$ function that satisfies these boundary conditions and Equation (11). Initially, all the inner nodes are assumed zero, and Equation (12) is applied over and over again, in an infinite loop. The iterations cause the boundary potentials to propagate node by node from both ends. As the number of iterations increases, the potential values at the inner nodes change because of the contributions from the left and right then converge. The difference between each iteration and the previous iteration becomes less and less, and finally settles to a fixed value. A kind of convergence criterion, such as "the iterations may be terminated if the sum of the differences between two consecutive iterations of all nodes is less than a given value," may be set. In one dimension, this is equivalent to setting

$$\sum_{k=1}^{KE}\left|U_k^{n+1} - U_k^n\right| \le \gamma, \tag{15}$$

where $\gamma$ is the total error limit. The arrows in Figure 2 show the propagation of the potential values at the nodes as the number of iterations increases. The thick line is the final potential distribution that satisfies a given error criteria.

Figure 3 shows the progress of iterations for the Poisson equation for the same space and boundary conditions given in Figure 2, but with a Gaussian type of charge distribution. The dots show the given charge distribution. The left and right vertical axes correspond to potential and normalized charge densities (i.e., $\rho_l/\varepsilon$), respectively. Figure 4 shows another simulation result for

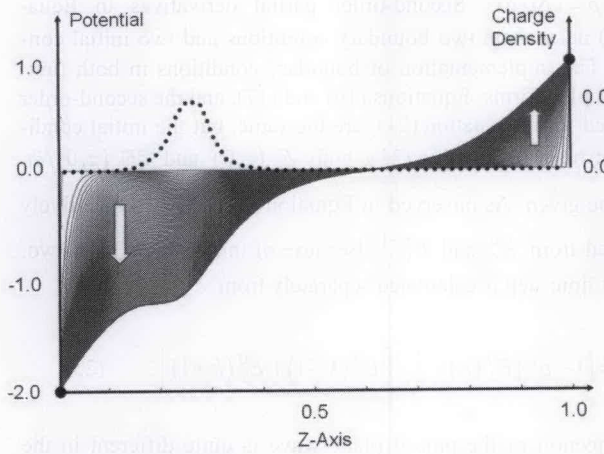


**Figure 3. A numerical solution of the Poisson equation for $Z_m = 1$, $KE = 100$, $U(0) = -2$, $U(1) = 1$, $\gamma = 146$, $\gamma \le 10^{-3}$. The dots show the charge distribution.**

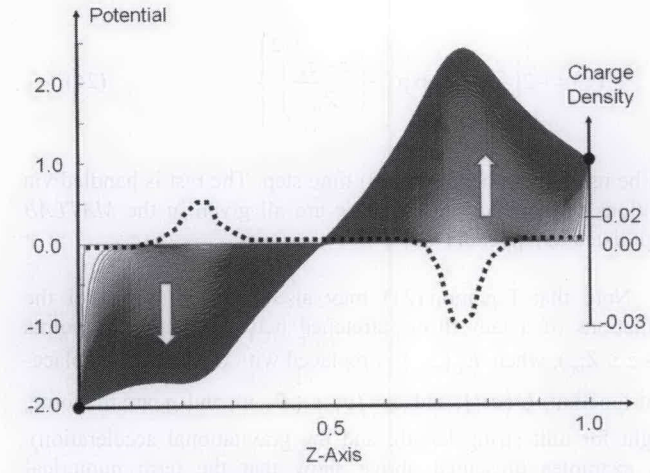


**Figure 4. The convergence of the Poisson equation for $Z_m = 1$, $KE = 100$, $U(0) = -2$, $U(1) = 1$, $\gamma = 256$, $\gamma \le 10^{-3}$. The dots show the charge distribution.**

the Poisson equation for the same parameters except the charge distribution. Again, the dots show the normalized charge distribution, and the left and right vertical axes correspond to the potential and charge densities, respectively. Finally, two examples for the solution of the Poisson equation in two dimensions, computed from Equation (14), are illustrated in Figure 5.

## 4.2 Wave Equation

Now, let us take the wave equation into account. Assume a plane wave, having $E_x(z,t)$ and $H_y(z,t)$, propagating along the $z$ direction in a finite region $0 \le z \le Z_m$ and for $t \ge 0$ (actually, the electric and magnetic fields have both $x$ and $y$ components in this case, but this assumption further simplifies the equations without loosing generality). The two partial differential (first-order coupled wave) equations under this assumption, derived directly from Maxwell's equations, are

$$\frac{\partial}{\partial z} E_x = -\mu_0 \frac{\partial}{\partial t} H_y, \tag{16}$$

$$\frac{\partial}{\partial z} H_y = -\varepsilon_0 \frac{\partial}{\partial t} E_x. \tag{17}$$

Note that the second-order uncoupled wave equation, Equation (10), is directly obtained from Equation (16) and (17) if, for example, one of them is further differentiated with respect to $z$ and they are combined to eliminate either $E_x(z,t)$ or $H_y(z,t)$. The unique solution of Equations (16) and (17) can be given when the initial and boundary conditions are specified.

The iterative equations for these coupled first-order plane-wave representations, discretized according to the finite-difference scheme, are

$$E_x^n(k) = E_x^{n-1}(k) - \frac{\Delta t}{\varepsilon_0 \Delta z}\left[H_y^n(k) - H_y^n(k-1)\right], \tag{18a}$$

$$H_y^n(k) = H_y^{n-1}(k) - \frac{\Delta t}{\mu_0 \Delta z}\left[E_x^n(k) - E_x^n(k-1)\right], \tag{18b}$$

where $z = k\Delta z$ and $t = n\Delta t$ ($k$ and $n$ are integers). Note that the classical Yee cell (leap-frog approach) is used here [2]. Therefore, the electric and magnetic fields along $z$ are located $\Delta z/2$ apart, and the magnetic fields are updated $\Delta t/2$ later than the electric fields. Electric (magnetic) fields in the next time step need only electric (magnetic) fields and two neighboring magnetic (electric) fields at the current time step.

Iterative equations are open-form representations, and are therefore conditionally stable: $\Delta t$ and $\Delta z$ can not be specified arbitrarily. The physical restriction of the stability condition is that the discrete time and spatial steps cannot be specified independently. Here, Courant stability applies: once $\Delta z$ is determined, $\Delta t$ is chosen such that $\Delta t \le \Delta z/c$. Finally, $\Delta z$ is determined according to the numerical dispersion effect: the highest frequency component should be discretized with a sufficient number of samples (e.g., $\Delta z \le \lambda_{min}/10$).

The source in Equation (18) can be injected in time using $E_x^n(k_s) = E_x^n(k_s) + g(n)$, where $g(n)$ is any suitable pulse function and $k_s$ is the source node. In general, the Gaussian pulse is used

$$g(n) = \exp\left\{-\left(\frac{n - n_0}{n_T}\right)^2\right\}, \tag{19}$$

with delay and pulse-width parameters $n_0$ and $n_T$, respectively. Pulse propagation at different instants of time along a 100-node discrete $z$ space with $k_s = 35$ and $n_T = 25$ is pictured in Figure 6. As observed, the injection starts as a kind of impulse, then a Gaussian-shaped pulse is formed and splits into two leftward ($-z$) and rightward ($+z$) propagating pulses. The space is terminated with a perfectly reflecting boundary at the left and right. The pulses are therefore totally reflected with the opposite phase when they hit the boundaries. (Note that a Gaussian pulse has low-pass filter characteristics: i.e., its frequency spectrum extends from dc to a maximum frequency that is inversely proportional to the pulse

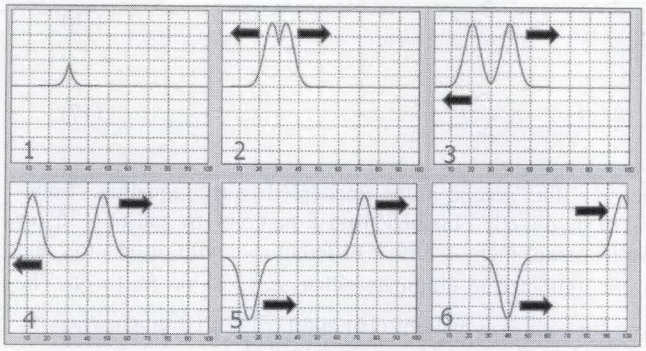


**Figure 6. Plane-wave pulse propagation at different time instants (the pulse was injected in time and PEC boundaries were used at the left and right ends).**

length. This is strange, because this pulse is propagating in air: dc cannot propagate, it can only be transmitted via transmission lines. This is why we call plane waves nonphysical, theoretical waves).

Alternatively, an initial spatial field distribution may be given. For example, the Gaussian-pulsed plane wave may be located in space before the time iteration starts:

$$E_x^0(k) = \exp\left\{-\left(\frac{k-k_s}{K_g}\right)^2\right\}, \tag{20a}$$

$$H_y^0(k) = \pm E_x^0(k)/\eta_0, \tag{20b}$$

$$\eta_0 = \sqrt{\mu_0/\varepsilon_0} = 377\,\Omega.$$

Here, $z_s = k_s \Delta z$ is the location of the pulse maximum, and $Z_g = K_g \Delta z$ is the extent of the spatial pulse. Note that a Gaussian pulse in space is injected into both the electric-field and magnetic-field components according to Ohm's Law. This guarantees one-way propagating pulsed plane-wave injection. The $\pm$ signs in Equation (20b) represent pulsed plane waves propagating towards the right and left, respectively. (Run the code listed in Appendix B with both signs separately and observe the right and left propagating waves injected via Equation (20). You'll see that the right-propagating pulsed plane wave is injected without any numerical noise, but the left propagating pulse has some residuals. Since iterations in the code are performed from left to right, the nature of a pulse propagating from left to right is coherent with the structure of the code.)

The second-order form for this plane-wave representation – for example, for the electric-field component – is

$$\left(\frac{\partial^2}{\partial z^2} - \frac{1}{c^2}\frac{\partial^2}{\partial t^2}\right) E_x(z,t) = 0. \tag{21}$$

The iterative form discretized according to the finite-difference scheme is

$$E_x^{n+1}(k) = 2\left(1-p^2\right)E_x^n(k) - E_x^{n-1}(k) + p^2\left[E_x^n(k) + E_x^n(k+1)\right] \tag{22}$$

where $p = c\Delta t/\Delta x$. Second-order partial derivatives in Equation (21) necessitate two boundary conditions and two initial conditions. The implementation of boundary conditions in both first-order coupled forms, Equations (16) and (17), and the second-order decoupled form, Equation (21), are the same, but the initial conditions are not: in Equation (22), both $E_x(z,0)$ and $\partial E_x(z,0)/\partial t$ should be given. As observed in Equation (22), $E_x^{n+1}$ is iteratively calculated from $E_x^n$ and $E_x^{n-1}$. Because of this, $n$ starts from two. The first time step is calculated separately from

$$E_x^1(k) = \left(1-p^2\right)E_x^0(k) + \frac{p^2}{2}\left[E_x^0(k-1) + E_x^0(k+1)\right]. \tag{23}$$

Time injection of the pulsed plane wave is quite different in the second-order wave equation. The pulsed plane wave may be injected from the left or right (i.e., as $E_x(0,t) = g(t)$) or $E_x(Z_m,t) = g(t)$). Alternatively, the pulsed plane wave may be located in space at the beginning. In this case, Equation (20a) will be used initially at the first time step, and the once-differentiated Gaussian function

$$g'(z) = -2\left(\frac{z-z_s}{Z_g}\right)\exp\left\{-\left(\frac{z-z_s}{Z_g}\right)^2\right\} \tag{24}$$

will be used in the next (second) time step. The rest is handled via iterations in Equations (22). These are all given in the *MATLAB* script listed in Appendix C.

Note that Equation (21) may also be used to model the oscillations of a taut string, stretched between two fixed points ($0 \le z \le Z_m$), when $E_x(z,t)$ is replaced with the vertical displacement function, $U(z,t)$, and $c = Tg/w$ ($T$, $w$, and $g$ are the force, weight for unit string length, and the gravitational acceleration). The examples presented above show that the term numerical propagation in electrostatics, in the iterative forms of the discrete Poisson and Laplace equations, refers to the convergence of the solution and not the electromagnetic wave propagation. On the other hand, in electrodynamics, numerical propagation refers to real wave propagation, since Maxwell's equations in their general form model electromagnetic waves.

## 4.3 *MATLAB*-Based *FDTD1D* Package

A nice educational package has been developed for the illustration of one-dimensional pulsed plane-wave propagation, and to observe its frequency spectrum. The front panel of the *MATLAB*-based *FDTD1D* virtual tool (visit http://www3.dogus.edu.tr/lsevgi for this and many virtual tools, previously published in the *Magazine*) is quite similar to those of the *TDRMeter* and *MGL-2D* virtual tools [3, 4]. The front panel of this virtual tool is given in Figure 7. The propagation medium may be free space, partially or fully dielectric-filled, finite, semi-infinite, or infinite. Perfectly reflecting and open boundary terminations may be chosen at the left. An impedance-type boundary condition is also implemented, in addition to these two, at the right termination.

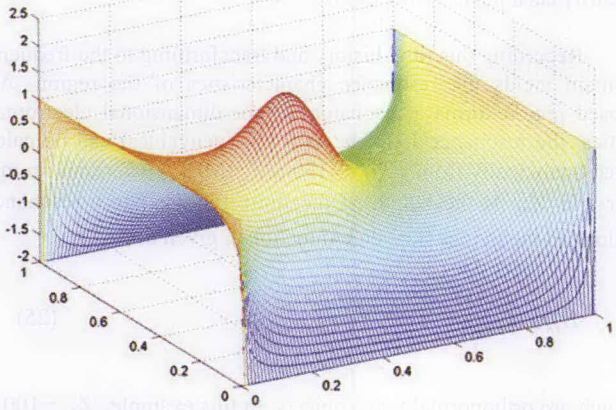

**Figure 5a. The solution of a two-dimensional Poisson equation for a 1 square meter plate ($100\times100$ space), for the boundary values of (left, right, top, bottom) $(1,\ 1,\ -2,\ -2)$, $\gamma = 942$, with similar Gaussian charge distributions inside ($\gamma \le 10^{-3}$).**

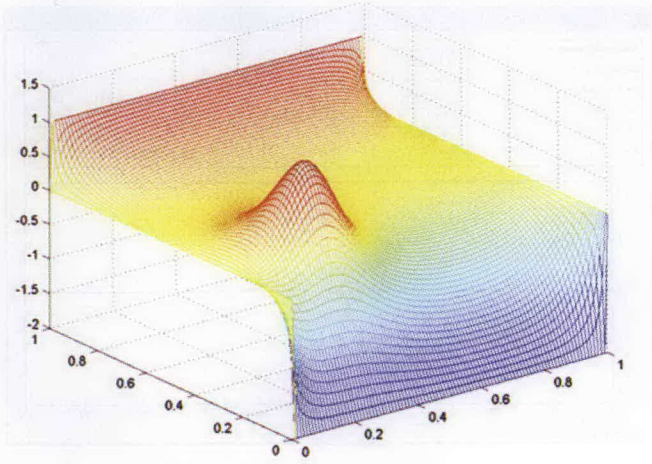

**Figure 5b. The solution of a two-dimensional Poisson equation for a 1 square meter plate ($100\times100$ space), for the boundary values of (left, right, top, bottom) $(0,\ 0,\ 1,\ -2)$, $\gamma = 865$, with similar Gaussian charge distributions inside ($\gamma \le 10^{-3}$).**

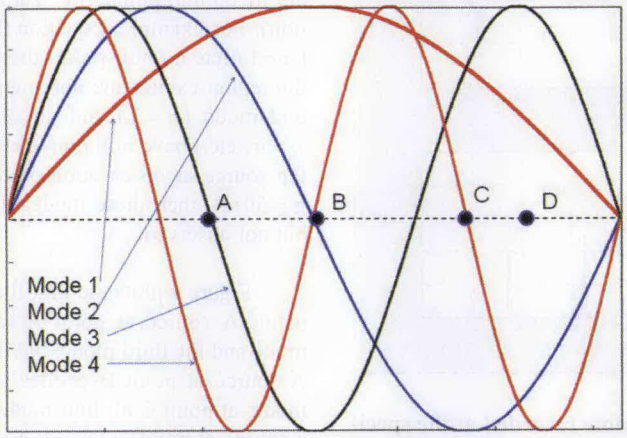


**Figure 9. The lowest four modes and their null points.**

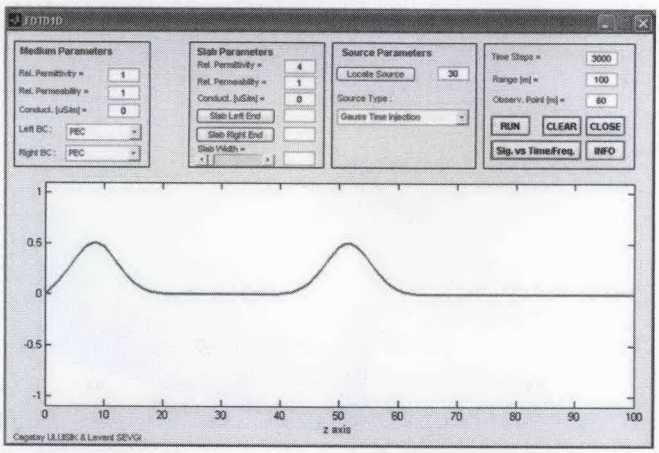


**Figure 7. The front panel of the *MATLAB*-based *FDTD1D* virtual tool.**

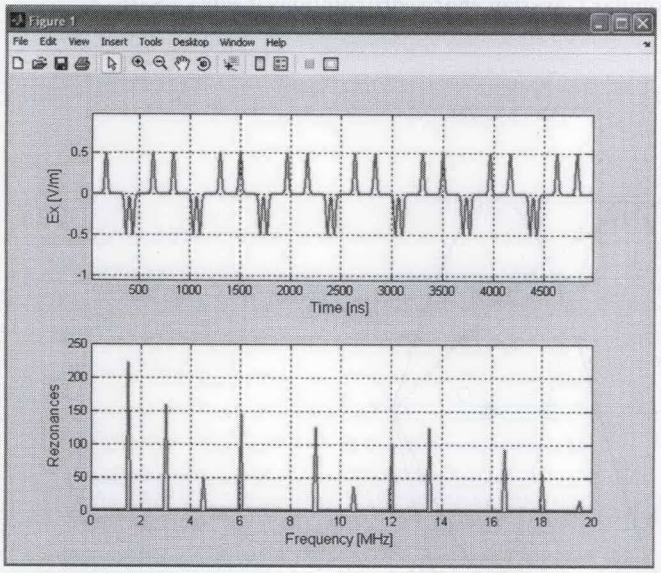


**Figure 8. The signal as a function of time recorded at the specified observation point and its frequency spectrum, showing the resonances.**

The front panel is divided horizontally into two. Parameters, grouped and located into four blocks, are supplied at the top; the electric field as a function of range is displayed at the bottom. The medium parameters are supplied in the first blocks. The user also specifies the termination conditions at the left and right within this block. The second-left block is reserved for the slab, which can be located vertically in the region, and the third block is used for the source specification. The source may be a Gaussian-pulse injection at anywhere in the region, a plane wave entering from left or right, or CW or rectangular-pulse injections. Finally, the last parameter block is for the operational parameters.

The `Plot Sg.vs.Time` button is used for offline separate graphs of signal as a function of time and its spectrum (i.e., signal as a function of frequency) at the chosen observation point. An example is given in Figure 8. The region was 100 m long, so the time step was nearly 3.3 ns. It took a pulse 330 ns to travel from one end to the other. The first pulse belonged to the pulse that directly traveled between the source and observer. The others were once-reflected, twice-reflected, etc., pulses. The reader may easily identify each pulse in this figure.

Recording this time history and transforming to the frequency domain yields the resonance characteristics of the region. Any closed region forms a resonator. In one-dimensional electromagnetics, the resonance frequencies (i.e., eigenvalues) can be calculated analytically from $(f_r)_n = 150n / Z_m$ [MHz], where $n$ is an integer and $Z_m$ is the length of the finite region. These resonances belong to the modes (i.e., eigenfunctions), given as

$$E_{xn}(z) = \sqrt{\frac{2}{Z_m}} \sin\left(\frac{n\pi}{Z_m} z\right), \tag{25}$$

which are orthonormal wave objects. In this example, $Z_m = 100$ m, and therefore the first (dominant) resonance frequency was 1.5 MHz, and the rest were integer multiples (i.e., 3.0 MHz, 4.5 MHz, 6.0 MHz, 7.5 MHz, etc.). The bottom plot in Figure 8 shows this spectrum. As observed, the resonances at integer multiples of 7.5 MHz were missing (see [5] for the mathematical and numerical requirements for the discrete and fast Fourier transformations, DFT, FFT).

In order to observe a resonance (an eigenvalue) in the frequency spectrum, both the source and the observation points *must not* lie on null points (i.e., zero crossings) of that mode (eigenfunction). For example, Equation (25) tells us that the dominant mode ($n = 1$) has no null point; therefore, a source located anywhere in the region excites the dominant mode. On the other hand, the second mode ($n = 2$) and all other even-order modes (i.e., $n = 4$, $n = 6$, etc.) have null points at mid-range. This means that if either the source or observation point is located at the midpoint (i.e., at $z = 50$ m), then these modes either are not excited, or are excited but not observed.

Figure 9 plots the first four modes of a one-dimensional resonator. A source at point A (at mid-range) excites the dominant mode and the third mode; but even-order modes can not be excited. A source at point B excites all four modes, except for the third mode; at point C all four modes but the fourth are excited. Finally, a source at point D excites all of them. In the above example, the locations of the source and observer were 30 m and 60 m, respectively. This means the source was located at the null point of the tenth mode and its integer multiples (twentieth, thirtieth, fortieth, etc.). The observer was located at nulls of fifth mode and its integer multiples. Therefore, the fifth, tenth, fifteenth, etc., resonances cannot be observed in the frequency spectrum.

## 5. Conclusions

Wave propagation and numerical propagation may sometimes be used improperly. People who deal with simulations in electrostatics and electrodynamics might refer to different concepts with the same term: *numerical propagation*. This is because the word *iteration* has different meanings in the numerical simulation of the Poisson and wave equations. It refers to the *convergence of the data* for the Poisson equation, but yields the *time progress* in the wave equation. This tutorial was devoted to the discussion of these terms. Numerical simulations of the Poisson/Laplace and wave equations were taken into account, and these terms were discussed for the discrete iterative solutions of these equations. *MATLAB* scripts were also supplied.

## 6. Acknowledgement

The author would like to express his gratitude to Çagatay Uluışık for his contributions in the preparation of the graphical user interface of the *MATLAB* package.

## 7. Appendix A: A Short *MATLAB* Script for the One-Dimensional Poisson Equation

```
% ======= Poisson_1D.m, Dec 2007 =========
clc; clear all; close all;

LZ = 1;      % input('Length of 1D Medium [m] : ?');
NZ = 100;   % input('Number of nodes along Z : ?');
NI = 1000;  % input('Number of iterations (NI) : ');
u(1) = input('Potential at left boundary (z=0): ');
u(NZ) = input('Potential at right boundary (z=LZ): ');

% Parameters
Eps=1.e-3; dz=LZ/(NZ-1);
i0=NZ/4; a0=1/sqrt(2.*pi*i0^2);

for i=1:NZ   % set initial values for z() and u()
   z(i)=(i-1)*dz;
   u(i)=0; Ui(i)=a0*exp(-(i-i0)^2/(NZ/4)));
end

figure (1); hold on; plot(z,u); xlim([0 LZ]);
xlabel('Z-axis'); ylabel('Potetial [V]');

% Main iteration loop
k=0; ueps=10;
while ueps>Eps
  k=k+1;  ueps=0;
  for i=2:NZ-1
     old=u(i);
     u(i)=(u(i+1)+u(i-1)-Ui(i))/2;
     new=u(i);
     ueps=ueps+(abs(new-old))^2;
  end
  if k>NI
   display('Maximum iteration number is exceeded !..');
break
  end
  plot(z,u);  pause(0.002);
end  % End of iteration loop
%-----------------------------
plot(z,u,'r', 'Linewidth', 2);
fprintf('Total Number of Iterations = %3d ', k);

figure(2);
subplot(2,1,1)
plot(z,Ui,'k--', 'Linewidth', 3); ylabel('Charge distribution')
subplot(2,1,2)
plot(z,u,'r', 'Linewidth', 2); xlabel(ZX-axis'); ylabel('Potetial
[V]');
```

## 8. Appendix B: A Short *MATLAB* Script for One-Dimensional First Order Coupled Wave Equations

```
% ======= FDTD1D.m, Dec 2007 =========

eps0=8.82e-12;  mu0=pi*4e-7;
eta=sqrt(mu0/eps0);
c=1./sqrt(eps0*mu0);
Exoldr=0; Exoldl=0;
rbc=0;          % Reflecting boundary at right
lbc=1;          % Open-boundary at left

nt=500;   %input('Number of time steps = ? ');
zmax=1;  %input('Maximum distance [m] = ? ');
ke=101;  %input('Number of range steps, ke [] = ? ');
kc=30;    %input('Source location, kc [] = ? ');

delz=floor(zmax/ke); dt=delz/c; tt=ke/2;   t0=10;
ce=dt/(eps0*delz);  ch=dt/(mu0*delz);

band=c/(10*delz);  alfa=3.3*band*band;
shift=4./sqrt(alfa);

% (1) put zeros (2) Inject a plane wave
for k=1:ke   %  Initial values along z
    Ex(k)=0.0;  Hy(k)=0.0;  z(k)=k;   % (1)
%  Ex(k)=exp(-(kc-k)^2/tt); Hy(k)=exp(-(kc-k)^2/tt)/eta;
end

t=0;
for n=1:nt   % FDTD time loop starts
   t=n*dt;
   for k=2:ke-1     % Ex is calculated alon z
      Ex(k)=Ex(k)+ce*(Hy(k-1)-Hy(k));
   end
% Inject Gaussian pulse
   Ex(kc)=Ex(kc) + exp(-alfa*(t-shift)^2);

   if lbc==0;  Ex(1)=Exoldl;   end
   if rbc==0; Ex(ke)=Exoldr; end

   for k=1:ke-1   % Hy is calculated along z
      Hy(k)=Hy(k)+ch*(Ex(k)-Ex(k+1));
   end
   Exoldr=Ex(ke-1);
   Exoldl=Ex(2);

   plot(z, Ex);
   xlim([1,ke]); ylim([-1.,1.]); xlabel('z axis'); grid;
   pause(0.001)
end % end of time loop
```

## 9. Appendix C: A Short Matlab Script for the One-Dimensional Second-Order Decoupled Wave Equation

```
% ======== WAVE1D.m, Dec 2007 ==========
% Eq: C^2 * u_z - u_tt = 0,   0 < z < Zm,   t > 0
% IC: U(z,0) = 0,   U_t(z,0) = 0
% BC: U(0,t) = f1(t), U(Zm,t) = f2(t) (Once-Diff. Gauss)
% ====================================
eps0=8.82e-12;  mu0=pi*4e-7;
eta=sqrt(mu0/eps0);  c=1./sqrt(eps0*mu0);

Zm = 1;          % input('Length of z space,  Zm [m] : ');
NZ = 100;        %input('Number of nodes, n [ ] : ');
kmax =  500;   %input('# of time iterations : ');

% Calculate other parameters (using stability condition)
z = 0:delz:Zm; delz=Zm/(NZ-1);
delt=delz/c;    p=c*delt/delz;

% Prepare 1D arrays for the iterations
u = zeros(1,NZ);        % Current time values
ue =  zeros(1,NZ);      % Previous time values
uy =  zeros(1,NZ);      % Next time values
uold=0.0;

% Source parameters
band=c/(10*delz);   alfa=3.3*band*band;
shift=4./sqrt(alfa);

% Locate Gaussian pulse in space (at t=0)
ns=NZ/4;   nT=NZ/5;
for k = 2:NZ-1
   f1=exp(-(k-ns)^2/nT);   u(k)=f1;
end

for k=2:NZ-1   % First Time step
   f2= -2*(k-ns)*exp(-(k-ns)^2/nT)/nT;
   uy(k)=(1-p^2)*u(k)+delt*f2+0.5*p^2*(u(k-1)+u(k+1));
end
ue=u;     u=uy;

for n=1:Nmax   % Time Iteration starts
   t=n*delt;
% Inject f1(t) from left or f2(t) from right
% u(1)=-(0.5*n/nt0)*exp(-16.*(n*delt-
                        nt0*delt)^2/(nT*delt)^2);
% u(n)=(0.25*n/nt0)*exp(-16.*(n*delt-
                        nt0*delt)^2/(nT*delt)^2);
 u(1)=0.0;   u(NZ)=0.0; % uold;
 for k=2:(NZ-1)
    uy(k)=2*(1-p^2)*u(k)-ue(k)+p^2*(u(k-1)+u(k+1));
 end
 uold=u(k-1);

% Inject source in time
% uy(ns)=uy(ns)-(0.5*n/nt0)*exp(-16.*(n*delt-
                        nt0*delt)^2/(nT*delt)^2);
 ue=u;  u=uy;    % Replace the arrays

% Plot wave vs. x-axis
 plot(z,u); xlim([1,Zm]);   xlabel('x axis'); pause(0.001)
end   % End of the time loop
```

## 10. References


1. C. F. Gerald and P. O. Wheatley, *Applied Numerical Analysis, Sixth Edition*, New York, Addison-Wesley, 1999.

2. K. S. Yee, "Numerical Solution of Initial Boundary Value Problems Involving Maxwell's Equations," *IEEE Transactions on Antennas and Propagation*, **AP-14**, May 1966, pp. 302-307.

3. L. Sevgi and Ç. Uluışık, "A *MATLAB*-Based Transmission-Line Virtual Tool: Finite-Difference Time-Domain Reflectometer," *IEEE Antennas and Propagation Magazine*, **48**, 1, 2006, pp. 141-145.

4. G. Çakır, M. Çakır, and L. Sevgi, "A Multipurpose FDTD-Based Two-Dimensional Electromagnetic Virtual Tool," *IEEE Antennas and Propagation Magazine*, **48**, 4, 2006, pp. 142-151.

5. L. Sevgi, "Numerical Fourier Transforms: DFT and FFT," *IEEE Antennas and Propagation Magazine*, **49**, 3, 2007, pp. 238-243.