Electrostatics with partial differential equations
– A numerical example

28th July 2011

This text deals with numerical solutions of two-dimensional problems in electrostatics. We begin by formulating the problem as a partial differential equation, and then we solve the equation by Jacobi's method. We use a package called Cython to improve computational speed.

1 Introduction

So far in the Electromagnetism course, we have looked at Coulomb's law, which describes the force between charged particles:

$$\mathbf{F} = \frac{1}{4\pi\varepsilon_0}\frac{q_1 q_2}{|\mathbf{r}|^3}\mathbf{r}. \tag{1}$$

We have seen that Coulomb's law can be derived from the more general Gauss' law, which is the first of the four Maxwell equations. Gauss' law states that the total *electrical flux* through a closed surface has to be proportional to the electric charge contained within the surface. Gauss' law can be expressed in integral form as follows:

$$\oint_C \mathbf{E} = \frac{Q}{\varepsilon_0} = \frac{1}{\varepsilon_0}\int_V \rho\, dV \tag{2}$$

where C is the surface encapsulatin the volume V, and ρ is the charge density. By applying the Divergence Theorem, we can write equation (2) on *differential form* as

$$\nabla \cdot \mathbf{E} = \frac{\rho}{\varepsilon_0} \tag{3}$$

Furthermore, we have defined *electric potential* by the relation

$$\mathbf{E} \equiv -\nabla V. \tag{4}$$

Inserting this relation into (3), we get

$$\nabla \cdot \mathbf{E} = \nabla \cdot (-\nabla V) = -\nabla \cdot \nabla V = -\nabla^2 V = \frac{1}{\varepsilon_0}\rho \tag{5}$$

where

$$\nabla^2 \equiv \nabla \cdot \nabla = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{6}$$

is the Laplace operator in two dimensions.

Now we have an equation relating the electrical potential in a point in space to the charge density in that point. This is a *partial differential equation*, which becomes clear if we write it out as

$$\frac{\partial^2 V(x,y)}{\partial x^2} + \frac{\partial^2 V(x,y)}{\partial y^2} = -\frac{1}{\varepsilon_0}\rho(x,y) \tag{7}$$

An equation on this form is known as *Poisson's equation*. If we are able to solve this equation for a given charge distribution, we know what the potential is anywhere in space. By taking the gradient of the potential we can find the electric field.

2 Point charge

We already know some solutions to this equation. For a point charge, the electric field will point radially outward (for a positive charge) or inwards (negative charge). The electric potential will form circular equipotential surfaces around the charge (spherical in three dimensions, of course). The reference value for the potential is arbitrary, and we usually define $V(\infty) = 0$. It is straightforward to find the electric field and the potential for one or several point charges by using Coulomb's law (1).

3 Numerical solution of Poisson's equation

In previous courses, you have worked extensively with ordinary differential equations (ODE), i.e. equations for functions of *one* variable. You have learned how to solve these by computer, using for instance Euler's method, Euler-Cromer or Runge-Kutta. If we are given an ODE with initial conditions, it is pretty straightforward to solve it numerically – it is just a question of computer power.

A partial differential equation (PDE) requires a bit more care. The solution we are after is a scalar field $V(x,y)$, assigning a value to every point on a two-dimensional plane. When we solve this equation numerically, we divide the plane into discrete points (i,j) and compute V for these points.

3.1 Dissecting the mathematics

We remember, e.g. from Euler's method for ODE's, that we solve our differential equation in discrete steps, and use the function's value computed in the previous step to find the value in the next step. With this in mind, it is reasonable to expect that we use the function values in neighbouring points to compute the function value for a PDE as well – and you will see that this is indeed the case. In this example we utilise what is called *Jacobi's method* to solve the Poisson equation for different charge distributions. Briefly explained, the method works as follows: We think of the xy-plane, divided into points, as a matrix. Our goal is to find a matrix V containing the values of V for all points (i,j), corresponding to discrete x and y values. We begin by guessing a matrix $V^{(0)}$, and use our PDE to calculate new values in a matrix $V^{(1)}$ based on the values from $V^{(0)}$, and so on. ((0) og (1) refer to the number of iterations, and must not be confused with exponents!) Hopefully, this process converges towards a matrix V that is a solution to our equation.

But how can we use Poisson's equation to calculate these values? First we need to do a few clever tricks.

The definition of the derivative is, as we know,

$$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \tag{8}$$

It can equivalently be written as

$$f'(x) = \lim_{h\to 0} \frac{f(x)-f(x-h)}{h} \tag{9}$$

where h is in the last term instead. If h is a small number, then

$$f'(x) \approx \frac{f(x+h)-f(x)}{h} \text{ or } f'(x) \approx \frac{f(x)-f(x-h)}{h}. \tag{10}$$

The definition of second derivative is in a similar way

$$f''(x) = \lim_{h\to 0} \frac{f'(x+h)-f'(x)}{h} \tag{11}$$

By inserting the last expression (9) for $f'(x)$ into the expression (11) for $f''(x)$, we get

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \tag{12}$$

For a function of two variables the procedure is the same, but we have to use partial derivatives. We write $\frac{\partial^2 V}{\partial x^2} \equiv V_{xx}$ to save some writing. We then get

$$V_{xx} \approx \frac{V(x+h,y) - 2V(x,y) + V(x-h,y)}{h^2} \tag{13}$$
$$V_{yy} \approx \frac{V(x,y+h) - 2V(x,y) + V(x,y-h)}{h^2} \tag{14}$$

By inserting this in equation (7) and solving for $V(x,y)$ (you should do this), we get

$$V(x,y) \approx \frac{1}{4}\left[V(x+h,y) + V(x-h,y) + V(x,y+h) + V(x,y-h)\right] + \frac{h^2}{4\varepsilon_0}\rho(x,y) \tag{15}$$

or, with discrete indices,

$$V_{i,j} = \frac{1}{4}\left[V_{i,j+1} + V_{i,j-1} + V_{i+1,j} + V_{i-1,j}\right] + \frac{h^2}{4\varepsilon_0}\rho_{i,j} \tag{16}$$

Keep in mind that indices in a matrix are reversed compared to what we are used to from coordinates – i describes which row we are in (y), and j describes which column (x).

This is the expression we need for Jacobi's method: By guessing an initial matrix $V^{(0)}$, we are able to find $V^{(1)}$ by using equation (16) and a computer loop.

But beware, there are some complications to this. If we look at equation (16), we see that we need to insert function values for all neighbouring points. This is fine as long as we are in the middle of the domain, but once we get to the edge we run into trouble. For instance, calculation of $V_{0,0}$ requires knowledge of $V_{-1,0}, V_{-1,-1}$ and $V_{0,-1}$, which are not included in our domain. Thus we do not have an expression for these boundary points, and we have to decide what values they should take prior to starting our algorithm. This is nothing unique to our method, but rather a mathematical consequence of working with a PDE. Our problem does not have a unique solution unless we specify the boundary vaules. This is why mathematical problems with PDE's are often called *boundary value problems*.

3.2 Code

Now that we have a machinery for solving our PDE numerically, it is time to put it to use. We begin by looking at a point charge as described above. Let us say we are looking at a square box with sides of length $L = 1$ (we use dimensionless units). We divide our box into $N = 21$ points and place our point charge in the middle of the box. We then get a charge density matrix ρ which is zero everywhere except in one point: $\rho_{11,11}$. Since we are dealing with discrete points, we can think of $(11, 11)$ as describing an area around the point, and $\rho_{11,11}$ as the charge density in this small (but finite) area.

Let us choose $\rho_{11,11} = 1$. We can always recalculate this to a realistic charge value later by multiplying the result by a factor. As mentioned, we have to guess an initial matrix $V^{(0)}$. A simple choice is the zero matrix. We can significantly lessen the time it takes to get convergence by making a clever guess – since the point charge problem is well-known we have a pretty good idea of what V is going to look like – but the point here is to let numerics do the job for us.

As for the boundary points, we will assume that the potential is zero along the edges. This is not unreasonable – remember that we usually define $V(\infty) = 0$, so if our domain is large then this is a good approximation.

We now have everything we need in order to write a program for solving the problem. We have chosen to write it in Python, example code below.

```
# point_charge.py - Iterative solution of 2-D PDE, electrostatics
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

# Set dimensions of the problem
L = 1.0
N = 21
ds = L/N

# Define arrays used for plotting
x = np.linspace(0,L,N)
y = np.copy(x)
X, Y = np.meshgrid(x,y)

# Make the charge density matrix
rho0 = 1.0
rho = np.zeros((N,N))
rho[int(round(N/2.0)),int(round(N/2.0))] = rho0

# Make the initial guess for solution matrix
V = np.zeros((N,N))


# Solver
iterations = 0
eps = 1e-8 # Convergence threshold
error = 1e4 # Large dummy error
while iterations < 1e4 and error > eps:
    V_temp = np.copy(V)
    error = 0 # we make this accumulate in the loop
    for j in range(2,N-1):
        for i in range(2,N-1):
            V[i,j] = 0.25*(V_temp[i+1,j] + V_temp[i-1,j] +
                    V_temp[i,j-1] + V_temp[i,j+1] + rho[i,j]*ds**2)
            error += abs(V[i,j]-V_temp[i,j])
    iterations += 1
    error /= float(N)
print "iterations =",iterations

# Plotting
matplotlib.rcParams['xtick.direction'] = 'out'
matplotlib.rcParams['ytick.direction'] = 'out'
CS = plt.contour(X,Y,V,30) # Make a contour plot
plt.clabel(CS, inline=1, fontsize=10)
plt.title('PDE solution of a point charge')
CB = plt.colorbar(CS, shrink=0.8, extend='both')
#plt.show()
```

If your try running this program, you will discover that it is a little slow. And we are running it with only 21×21 points – not a good resolution. So, we need to speed it up. There are several possible improvements that can be made, but perhaps the most obvious one is this:

Consider the (hopefully) well-known ODE algorithm Euler-Cromer's method (for second-order equations or higher). It differs from Euler's method in one way: When we have computed the value of the 1st order derivative in a step, this new value is used for computing the 0th order derivative – instead of the previous one as in Euler's method.

This simple idea can be translated to our PDE problem. The algorithm above runs through the points the same way we read text: It starts at the top left corner, and works its way horizontally until it hits the end, then continues from the left on the next row. This means that for all points above and to the left of the current point, we have updated values. If we tell the program to use these values instead of the old ones, we can, hopefully, get faster convergence. Implementing this simply means replacing `V_temp` with `V` in the calculation. The updated solver module is included below.

```
 # Solver - point_charge_improved.py
while iterations < 1e5 and error > eps:
    V_temp = np.copy(V)
    error = 0 # we make this accumulate in the loop
    for j in range(1,N-1):
        for i in range(1,N-1):
            V[i,j] = 0.25*(V[i+1,j] + V[i-1,j] +
                    V[i,j-1] + V[i,j+1] + rho[i,j]*ds**2)
            error += abs(V[i,j]-V_temp[i,j])
    iterations += 1
print "iterations =",iterations
```

But alas, this still is not very fast. There are other ways of improving the algorithm, but they are not really worth the trouble. *The main bottleneck here is Python.* Python is a *high-level language*, meaning that it is easy to write and understand, but comes with lots of built-in systems to protect the user from having to think of everything. These systems produce what is known as *overhead*, and this greatly slows down Python. For this reason, professional simulations are always done in *low-level languages* such as C, C++ or Fortran. These are much faster, but more difficult to use.

4 Introducing: Cython

But luckily, there are brilliant people out there, and a few of them have taken the trouble to write a Python package known as *Cython*. Cython provides a link between Python and C, making it possible to utilise some of C's fastness without actually writing C. The next portion of this text deals with implementing our code in Cython. This will give us a Python program capable of handling quite big problems.

4.1 Installing Cython

The first thing we need to do is install Cython. We assume you are using a Linux computer at UiO with Python installed. (If you are using your personal computer with Ubuntu, look further down.) Begin by pasting the following lines into your terminal and executing them, one at a time:

```
cd ~/
wget http://cython.org/release/Cython-0.14.1.tar.gz
tar xzvf Cython-0.14.1.tar.gz
cd Cython-0.14.1
python setup.py install
```

This will install Cython to a directory in your home folder. You should get an error message during the installation, since you don't have write access to the directory where Python is installed. This requires a little tweak to get around. Paste and execute the following lines:

```
cd ~/
echo "export PATH=~/Cython-0.14.1/bin:$PATH" >> .bashrc
echo "export PYTHONPATH=~/Cython-0.14.1:$PYTHONPATH" >> .bashrc
source .bashrc
```

This adds two lines to your terminal shell configuration and reloads the configuration file, to make bash tell Python that Cython is installed. And with that, we are good to go.

Personal Ubuntu computer: *The procedure is very similar, but since you do have write access you won't need the .bashrc tweak. You just have to make sure that you run the install command as super user. So instead of* `python setup.py install`*, you should run*

```
sudo python setup.py install
```

Some people have reportedly run into problems when trying to install Cython on Ubuntu with this approach. You may have to install the Python developer version, which you can get in apt by the command

```
sudo apt-get install python-dev
```

4.2 Implementing our code in Cython

What we are looking for is increased speed, but we do not want to write C, or having to make extensive modifications to our Python program. So we consider the program, and realize that almost all runtime is spent in the `while` loop. If we can optimize the loop to make it run with C, then we should get much more speed. Below is the modified Python program to this effect. The changes are highlighted.

```
# point_charge.pyx - Iterative solution of 2-D PDE, electrostatics
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Cython-specific imports
cimport numpy as np
cimport cython
from libc.math cimport sqrt
ctypedef np.float_t FTYPE_t

@cython.boundscheck(False)
def main():
    # Declare variables
    cdef int N, iterations, i, j
    cdef double L, ds, rho0, eps, error
    cdef np.ndarray[FTYPE_t] x, y
    cdef np.ndarray[FTYPE_t, ndim=2] X, Y, rho, V, V_temp

    # Set dimensions of the problem
    L = 1.0
    N = 101
    ds = L/N

    # Define arrays used for plotting
    x = np.linspace(0,L,N)
    y = np.copy(x)
    X, Y = np.meshgrid(x,y)

    # Make the charge density matrix
    rho0 = 1.0
    rho = np.zeros((N,N))
    rho[int(round(N/2.0)),int(round(N/2.0))] = rho0

    # Make the initial guess for solution matrix
    V = np.zeros((N,N))


    # Solver
    iterations = 0
    eps = 10**(-8.0) # Convergence threshold
    error = 10**(4.0) # Large dummy error
    while iterations < 1e6 and error > eps:
        V_temp = np.copy(V)
        error = 0 # we make this accumulate in the loop
        for j in range(1,N-1):
            for i in range(1,N-1):
                V[i,j] = 0.25*(V[i+1,j] + V[i-1,j] +
                        V[i,j-1] + V[i,j+1] + rho[i,j]*ds**2)
                error += sqrt((V[i,j]-V_temp[i,j])*(V[i,j]-V_temp[i,j]))
        iterations += 1
        error /= float(N)
    print "iterations =",iterations

    # Plotting
    matplotlib.rcParams['xtick.direction'] = 'out'
    matplotlib.rcParams['ytick.direction'] = 'out'
    plt.title('Point Charge')
    CS = plt.contour(X,Y,V,30)
    CB = plt.colorbar(CS, shrink=0.8, extend='both')
    plt.clabel(CS, inline=1, fontsize=10)
    plt.show()
main()
```

So, what has changed? Starting from the top, we see that we have some additional import statements. `cimport` is the Cython import statement, and we need numpy and, of course, Cython, to be imported and understood by the C compiler. The line `from libc.math cimport sqrt` imports C's square root function from the C library – since this function is used in every iteration, it's a good idea to use a fast implementation of it. If, later, you wish to experiment with Cython for other applications, then you may need to explore other functions to be imported this way (e.g. the exponential function `exp`).

Directly below is the line `ctypedef np.float_t FTYPE_t`. This specifies to Cython that we are going to be defining arrays of floats, not integers or strings.

The next line is a little cryptic: `@cython.boundscheck(False)`. This is the magical command. It tells Cython not to bother checking our code for errors (like trying to index lists beyond their length). With Python, such an error produces an error like `Index out of bounds`. With C, all you get is `Segmentation fault`. C knows that something went wrong, but it has not bothered to find out what. It does not help us with debugging, but when we have a working script it greatly reduces the *overhead* discussed above.

The observant reader will notice that we have put our program inside a function `main()`, which is then called at the end of the script. This is simply a workaround neccessary to use Cython. The program works exactly the same.

The last changes are the four lines of `cdef int N, ...`. This is essential! Here we list all variables used in our program, and specify what kinds of objects they are going to be. The first line are the *integers* (notice the counter variables i and j, they are integers too). *Double* is C-language for *float*. `np.ndarray` specifies an array, and here we use the `FTYPE_t` definition from above to specify float. The last line defines matrices, distinguished by `ndim=2`.

4.3 Running Cython

To use our Cython program, it needs to be *compiled*. Begin by saving your Cython script (you may just copy-paste the one above) as a `.pyx` file (as opposed to `.py`!). Then, open an empty document named `setup.py` and paste the following code into it:

```
# setup.py
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

numpy = "/local/lib/python2.5/site-packages/numpy/core/include/"

setup(
    cmdclass = {'build_ext': build_ext},
    ext_modules = [Extension("point_charge_lib", ["point_charge.pyx"],
                             include_dirs = [numpy])]
)
```

You should replace the file name (“point_charge.pyx”) with the name of your Cython program. The library name (“point_charge_lib”) is arbitrary – you can call it whatever you want – but it should be descriptive. (For the record: The name “setup.py” is arbitrary too, and if your make multiple Cython scripts you might want to have designated setup files – “setup_point_charge.py”, etc.)

Now we are ready to compile. This is done by one line (you obviously have to be in the directory where you saved your files):

```
python setup.py build_ext --inplace
```

With this, Cython translates our Python-esque code into C. Our program can now be run by the command

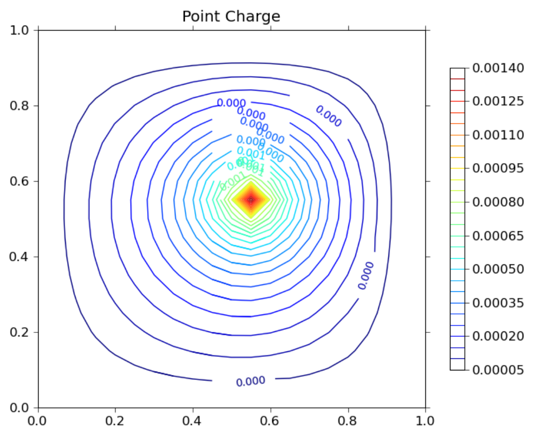
```
python -c "import point_charge_lib"
```

where “point_charge_lib” is your library name from above.

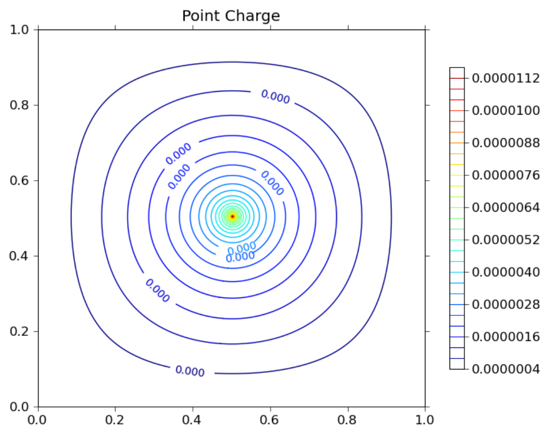
And that’s it! We are now running C without running C. (Or at least without having to think about it.)

5 Analyzing our data

Now that we (finally) have our full machinery up and running, it is time to do some physics. If you have tried to run the Cython program for the point charge, you should get this plot:

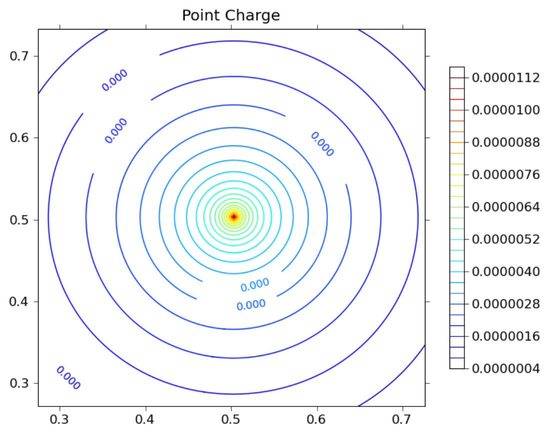


The first thing to notice is the low resolution. The edges of the contour are jagged, and the red area close to the charge is diamond-shaped. Not good. By changing the value of N in our Cython script to $N = 201$ (remember to compile again!), this problem is amended (a resolution as large as this is the reason we are running Cython, as Python would take forever):



The second thing to notice, which is apparent in both plots, is that the contour lines close to the edges are not circular. This is due to a problem with our setup: The point charge problem has circular symmetry. We are using a square box, and have defined the potential to be zero around the edges. **The results close to the edges are unphysical.** Well, not unphysical per se, but they correspond to having the charge inside a grounded box – not in free space, as we wanted. To get it right, we should have used a circular domain.

But the problem only exists close to the edges. Far from the edges, the solution does not depend heavily on the boundary values, and should be usable. We can easily examine this by zooming the plot a bit. To zoom in matplotlib, click the button looking like a cross – this is the pan/zoom utility. Place your mouse pointer on the red dot in the centre, click and hold your right mouse button and drag the pointer upwards and to the right. Zooming beyond the outmost two contour lines, it is looking a lot better:



5.1 Other applications

The point charge is just one example of an electrostatic problem. Another commonly encountered problem is the *capacitor*: Two plates with opposite charge placed close together. Our code is easily modified to study other problems – all we need to do is edit the charge density matrix. Here is a suggestion for modelling the capacitor (to replace the line `rho[int(round(N/2.0)),int(round(N/2.0))] = rho0`):

```
for j in range(round(N/2.0)-int(N/20.0),round(N/2.0)+int(N/20.0)):
        rho[round(N/2.0)-int(N/30.0),j] = rho0
        rho[round(N/2.0)+int(N/30.0),j] = -rho0
```

You should play around with other setups, e.g. a dipole, multiple poles, random charge distributions, etc.

Solving the Generalized Poisson Equation Using the Finite-Difference Method (FDM)

James R. Nagel, nageljr@ieee.org
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, Utah

February 15, 2012

1 Introduction

The Poisson equation is a very powerful tool for modeling the behavior of electrostatic systems, but unfortunately may only be solved analytically for very simplified models. Consequently, numerical simulation must be utilized in order to model the behavior of complex geometries with practical value. Although there are several competing algorithms for achieving this goal, one of the simplest and more straightforward of these is called the *finite-difference method* (FDM). At its core, FDM is nothing more than a direct conversion of the Poisson equation from continuous functions and operators into their discretely-sampled counterparts. This converts the entire problem into a system of linear equations that may be readily solved via matrix inversion. The accuracy of such a method is therefore directly tied to the ability of a finite grid to approximate a continuous system, and errors may be arbitrarily reduced by simply increasing the number of samples.

Despite the relative simplicity of FDM as a numerical tool, information on the subject is surprisingly scarce. This is especially true for the case of quasi-static systems experiencing current flow in conducting materials. Part of the reason for this likely has to do with the fact that FDM is, at its core, nothing more than a simplified form of the finite element method (FEM). The only difference is that FDM is solved through a fixed, rectangular geometry, while FEM utilizes a flexible, triangular mesh. Nevertheless, the uniform grids inherent to FDM make it very intuitive to learn and to program, especially for students unfamiliar with techniques in numerical methods. Consequently, the learning curve for FEM is far steeper than it is for FDM, and often requires a whole semester of study to fully understand. On the other hand, expertise with FDM may be readily achieved in only a few weeks, and even serves as an intuitive springboard from which to study the more complex nature of FEM.

The goal of this paper is to serve as a comprehensive introduction to the principles of FDM. Much of the basic information is readily found in standard textbooks [1, 2], though many of the practical details and advanced topics are difficult to find anywhere at all. This paper is therefore a compilation of knowledge based on my experience with FDM, as well as a primer on some of the more advanced topics that are almost nonexistent in the literature. The audience is specifically intended to include first-year students in computational electromagnetics, but more advanced professionals should still find useful reference material as well. The basic governing equations are derived directly from Maxwell's equations and FDM is first introduced in its most basic formulation. The algorithm is then extended from the classical Poisson equation to the generalized Poisson equation in order to include the effects of varying dielectrics within the domain. Finally, we conclude with an extension of FDM to include quasi-static systems by showing how the exact same governing equation still applies for complex-valued phasors.

2 The Generalized Poisson Equation

Beginning with Maxwell's equations, the ultimate governing equation for any electrostatic system is Gauss's law. Expressed in point form, this may be written as

$$\nabla \cdot \mathbf{D}(\mathbf{r}) = \rho(\mathbf{r}) \ . \tag{1}$$

In this context, $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$ is a position vector in space, ρ is the charge density function, and $\mathbf{D}$ is the electric flux density. Using the constitutive relation $\mathbf{D}(\mathbf{r}) = \epsilon_0\epsilon(\mathbf{r})\mathbf{E}(\mathbf{r})$, Gauss's law may be rewritten in terms of the electric field intensity $\mathbf{E}$ as

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \mathbf{E}(\mathbf{r})\Big] = \frac{\rho(\mathbf{r})}{\epsilon_0} \ , \tag{2}$$

where $\epsilon(\mathbf{r})$ is the dielectric constant as a function of position in space and $\epsilon_0 = 8.854 \times 10^{-12}$ F/m is the permittivity of free-space. Gauss's law may be further rewritten in terms of the *voltage potential function* $V(\mathbf{r})$ by making the substitution $\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r})$:

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] = -\frac{\rho(\mathbf{r})}{\epsilon_0} \ . \tag{3}$$

Although it is not commonly discussed in the literature, this is really nothing more than a generalized form of the *Poisson equation*, and is the expression we shall be most interested in throughout this paper. A far more familiar expression occurs if we next assume a uniform dielectric function with the form $\epsilon(\mathbf{r}) = \epsilon_r$. This gives us

$$\nabla^2 V(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0\epsilon_r} \ , \tag{4}$$

which is the classical form for the Poisson equation as given in most textbooks.

Although the classical Poisson equation is much simpler to numerically solve, it also tends to be very limited in its practical utility. Realistically, the generalized Poisson equation is the true equation we will eventually need to solve if we ever expect to properly model complex physical systems. We shall therefore begin by using the classical Poisson equation as a demonstration case for how FDM works before expanding our algorithm to the generalized form. For brevity and simplicity, this paper will be strictly limited to two-dimensional systems, though a full three-dimensional solution follows a nearly identical derivation.

3 The Five-Point Star

The first step in applying FDM is to define a *mesh*, which is simply a uniform grid of spatial points at which the voltage function will be sampled. Letting h be the distance between each sample, the points that lie on the mesh may be defined by

$$x_i = ih \ , \quad \text{and} \tag{5}$$

$$y_j = jh \ , \tag{6}$$

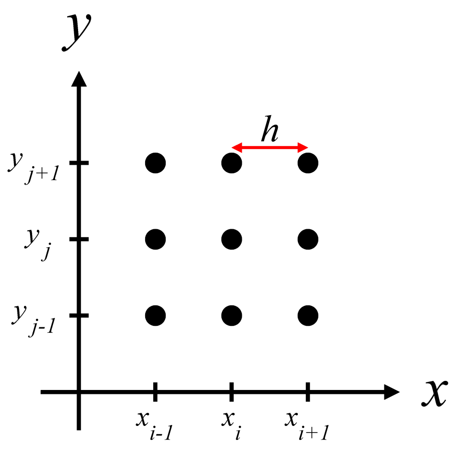


Figure 1: Mesh points for the FDM grid.

where i and j are integers. In practice, i and j will eventually be used as indices for a matrix of voltage samples, so it helps to use a short-hand notation that bears this in mind. We shall therefore replace the spatial coordinates with simple indices by assuming the following convention:

$$V(i,j) = V(x_i, y_j) \ . \tag{7}$$

In a similar fashion, we may also define the charge density samples along the same mesh by using the $\rho(i,j)$ notation.

The next step is to expand the Poisson equation by explicitly showing the partial derivatives in space:

$$\frac{\partial^2 V(i,j)}{\partial x^2} + \frac{\partial^2 V(i,j)}{\partial y^2} = -\frac{\rho(i,j)}{\epsilon_0} \ . \tag{8}$$

The reason for doing this is so that we may approximate the derivative operators through the use of finite-differences. The easiest way to do this is through the three-point approximation for the second-derivative, which is given as

$$\frac{\partial^2}{\partial x^2} V(i,j) \approx \frac{V(i-1,j) - 2V(i,j) + V(i+1,j)}{h^2} \ , \tag{9}$$

with a similar expression for the partial derivative with respect to y. Plugging back into Equation (8) then gives us

$$V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) - 4V(i,j) = -\frac{h^2}{\epsilon_0}\rho(i,j) \ . \tag{10}$$

Finally, we solve for $V(i,j)$ to find

$$V(i,j) = \frac{1}{4}\left[V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) + \frac{\rho(i,j)h^2}{\epsilon_0}\right] \ . \tag{11}$$

What this expression tells us is that every voltage sample $V(i,j)$ is dependent only on $\rho(i,j)$ and the voltage at the four nearest neighbors. A graphical depiction of this is called a *computational molecule*, and is shown in Figure 2. Because of its unique geometry, this five-point stencil is often referred to as the *five point star*.

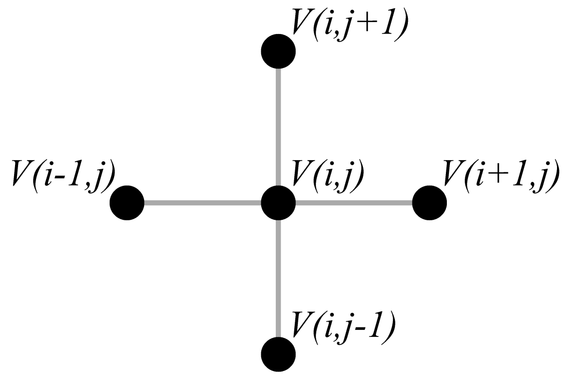


Figure 2: Computational molecule for the 5-point star.

Because each voltage sample $V(i,j)$ is linearly dependent on its four nearest neighbors, the solution over all (i,j) may be represented as a simple matrix-vector equation. This is readily achieved by defining the vector $\mathbf{x}$ to contain all of the voltage samples within the domain. For example, one simple method might scan row-wise along the voltage samples according to the convention:

$$\mathbf{x} = \left[\begin{array}{cccccccc} V(1,1) & V(1,2) & V(1,3) & \cdots & V(2,1) & V(2,2) & V(2,3) & \cdots \end{array}\right]^T . \tag{12}$$

The next step is to express the linear relationship between voltage samples into a matrix $\mathbf{A}$. This effectively converts the entire problem into a matrix-vector equation with the form

$$\mathbf{Ax} = \mathbf{b} , \tag{13}$$

where $\mathbf{b}$ contains all the information about any charge densities and boundary conditions. The numerical solution to the system is finally found by simply inverting the matrix $\mathbf{A}$ to arrive at

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} . \tag{14}$$

4 Boundary Conditions

Because a computer can only store a finite number of grid points, it is always necessary to truncate a simulation domain along some fixed boundary. Since the five-point star is not applicable at the boundary samples, it is necessary to specify boundary conditions in order to arrive at a unique solution to the problem. The two most basic forms of boundary condition are called the *Dirichlet* boundary and the *Neumann boundary.* In practice, it is common for simulations to employ a mixture of these two conditions at the edges, so it is helpful to define Ω_D and Ω_N as the set of all points which satisfy the Dirichlet and Neumann conditions.

The simplest boundary condition is the Dirichlet boundary, which may be written as

$$V(\mathbf{r}) = f(\mathbf{r}) \qquad (\mathbf{r} \in \Omega_D) . \tag{15}$$

The function f is a known set of values that defines V along Ω_D. Thus, the Dirichlet boundary is nothing more than a forced solution to the potential function at specific points. A good example of such a condition occurs in the presence of charged metal plates. Because all points on a metal are

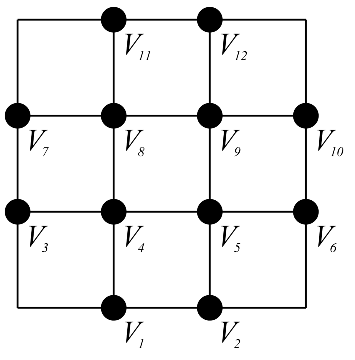


Figure 3: Sampled grid of voltages from Example 1.

at the same potential, a metal plate can readily be modeled by a region of points with some fixed voltage. In contrast, the Neumann boundary condition exists when the derivative of the potential function is known. Generally speaking, the derivative is defined with respect to the outward unit normal at the boundary, which is written as

$$\frac{\partial V(\mathbf{r})}{\partial \mathbf{n}} = f'(\mathbf{r}) \qquad (\mathbf{r} \in \Omega_N) \ , \tag{16}$$

where $\mathbf{n}$ is the outward-pointing unit normal vector, and f' the specifies the set of known derivatives. Unlike the Dirichlet condition, the Neumann does not offer a direct solution to the voltage potentials on a discrete, sampled grid. Rather, the boundary point must be expressed in terms of the surrounding points by applying a new stencil. The simplest method for expressing this is by imagining a central-difference approximation between the boundary sample V_b and the first inner sample V_i:

$$\frac{V_b - V_i}{h} = f' \ . \tag{17}$$

It is important to remember that the derivative function f' is defined with respect to the outward normal direction. This convention allows us to express Equation (17) the same way without any care for where exactly the boundary itself is located, be it top, bottom, left, or right of the simulation domain.

4.1 Example 1: A simple 4 × 4 grid

Consider the simple, 4 × 4 grid of voltage samples depicted in Figure 3. The top boundary is a Dirichlet boundary fixed at 1.0 V with bottom boundary grounded at 0.0 V. The left and right boundaries are Neumann boundaries fixed to a derivative of 0.0 V/m with respect to the outward normal. Using FDM, it is our job to solve for the voltage potentials at all of the indicated points.

The first step is to establish some sort of numbering convention so that the unknown vector $\mathbf{x}$ may be defined. One straightforward way to do this is by scanning across the rows, as indicated by the numbering in Figure 3. It is also worth emphasizing that the samples along the corners of the domain do not make any difference to the final solution of the problem with respect to the interior points. This is because neither the boundary conditions nor the five-point star will depend on what values are placed within the corners. We will therefore neglect these points entirely from the solution set, though in practice it can often be easier to just assign convenient values to them.[1] The vector of unknowns will therefore be written as

$$\mathbf{x} = \left[\begin{array}{ccccc} V_1 & V_2 & V_3 & \cdots & V_{12} \end{array} \right]^T .$$

The next step is to fill the system matrix **A**. We begin by noting that V_1 and V_2 are both Dirichlet boundaries fixed at 0.0 V. The first two rows in **A** are therefore nothing but zeros with a one placed at the diagonal element. The same is also true for V_{11} and V_{12} since these are likewise Dirichlet boundaries. The Neumann boundaries are filled in a similar manner, but with a -1 placed on the column corresponding to the interior point. For V_3 and V_7, this is the first element to the right of the diagonal. For V_6 and V_{10}, the -1 is placed at the first element to the left of the diagonal. For the remainder of the samples, the five-point star dictates a value of -4 to be placed at the diagonal, with four 1's placed at their corresponding columns that represent the neighboring points. This will include the two columns immediately adjacent to the diagonal, plus two other 1's placed at the appropriate locations.

The final step is to fill the forcing vector **b**. Generally speaking, this will be a vector of all zeros except at the points where there is a nonzero boundary condition or a nonzero value for ρ. Thus, **b** has only two 1's placed in the last two rows, with zeros placed at all other elements. Writing out the full linear system $\mathbf{Ax} = \mathbf{b}$ therefore leads to

$$\begin{bmatrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1}
\end{bmatrix}
\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ V_{11} \\ V_{12} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} .$$

Finally, we solve for $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ to find

$$\mathbf{x} = \left[\begin{array}{cccccccccccc} 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 1 & 1 \end{array} \right]^T \text{ Volts} .$$

5 Successive Over-Relaxation

For relatively small simulation domains, the direct matrix inversion demonstrated by the previous example works perfectly well for obtaining a solution. However, it is important to realize that the size of **A** grows directly with the *square* of the size of **x**. For example, given a rectangular simulation domain of 100×100 voltage samples, the matrix **A** will need to be $10,000 \times 10,000$ elements. Because direct matrix inversion is such an intense operation, it is easy to see how even small simulations can quickly require excessive computational resources.

To reduce the computational cost required by direct matrix inversion, it helps to realize that **A** is a *sparse* matrix, meaning the vast majority of elements in **A** are all zeros. This is a direct

[1]Imagine a square simulation grid of 200×200 voltage samples. Is it really worth the effort to write a specialized algorithm just to save memory on four measly samples?

consequence of Equation (11), which shows that each voltage element is only dependent on four other samples. As a result, each row in $\mathbf{A}$ has, at most, only five nonzero entries (or even one nonzero entry if the voltage sample is a fixed Dirichlet boundary). This allows us arrive at a solution through the use of sparse matrix solvers that take take advantage of this property. Although there are many available methods to chose from [3], we will focus on a very simple algorithm called *successive over-relaxation* (SOR).

The first step when utilizing SOR is to define the *residual* $R(i,j)$ as the degree to which each voltage sample $V(i,j)$ does not satisfy Equation (11):

$$R(i,j) = \frac{1}{4}\left[V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) + \frac{\rho(i,j)h^2}{\epsilon_0}\right] - V(i,j) \tag{18}$$

The next step is to loop over every sample in $V(i,j)$ and add a correction factor defined by $R(i,j)$. This process is then repeated over many iterations until the residual falls below some acceptable error value. For the kth iteration in the loop, we therefore have

$$V^{k+1}(i,j) = V^k(i,j) + R^k(i,j) \ . \tag{19}$$

This method is referred to as *successive relaxation*, and is guaranteed to eventually converge on the correct solution.

Although convergence is a desirable property of successive relaxation, a far more practical property is *rapid* convergence. This may be achieved if we first multiply R with a *relaxation factor* ω, such that

$$V^{k+1}(i,j) = V^k(i,j) + \omega R^k(i,j) \ . \tag{20}$$

This is called the method of successive *over*-relaxation, and will also converge as long as we enforce the condition that $0 < \omega < 2$. The only tricky part is choosing an ideal value for ω that minimizes the time to convergence. Generally speaking, this can only be found through empirical trial-and-error, but a special case arises for rectangular grids. The proof of this is rather involved, so we shall simply state the end result from [1] as

$$\omega = \frac{8 - \sqrt{64 - 16t^2}}{t^2} \ , \tag{21}$$

where N_x and N_y are the number of grid samples in the x- and y-directions, and

$$t = \cos(\pi/N_x) + \cos(\pi/N_y) \ . \tag{22}$$

6 Electric Fields

From the basic definition $\mathbf{E} = -\nabla V$, it is a straightforward process to extract electric fields from a complex simulation of voltage potentials. We therefore begin by expanding the definition of the gradient into its constituent x and y components:

$$\mathbf{E}(\mathbf{r}) = -\hat{\mathbf{x}}\frac{\partial V(\mathbf{r})}{\partial x} - \hat{\mathbf{y}}\frac{\partial V(\mathbf{r})}{\partial y} \ . \tag{23}$$

The next step is to apply the central difference approximation to the individual field components. The resulting grid of electric field samples is therefore slightly staggered from the voltage potentials

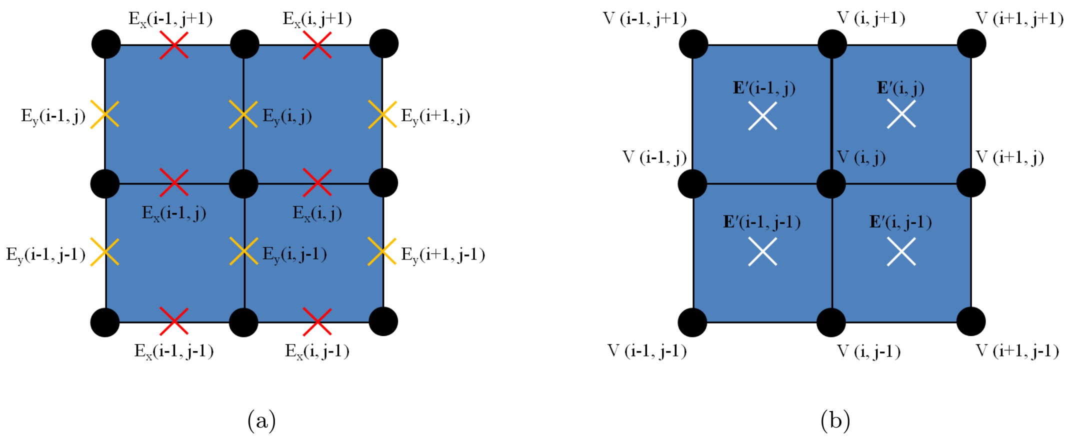


Figure 4: Grid stencil for obtaining the electric field samples. Circles represent voltage samples while $\times$'s represent the electric field samples. (a) Central differences are first calculated individually along the x and y directions, (b) then averaged together to produce a field sample that is staggered from the voltage grid.

according to the convention shown in Figure 4(a). These components are thus given as

$$E_x(i,j) = -\frac{V(i+1,j) - V(i,j)}{h} \ , \tag{24}$$

$$E_y(i,j) = -\frac{V(i,j+1) - V(i,j)}{h} \ . \tag{25}$$

At this point, two points of information are worth noting. The first is that the E_x grid contains one fewer sample along the x-direction than does the V grid. Similarly, the E_y grid is comprised of one fewer sample along the y-direction. This is simply a natural result of applying the central-difference method to compute a numerical derivative. The second point is that the x- and y-components of the electric field are staggered from each other in space. In order to fix this, we must apply a second approximation by averaging the field components together according to the geometry in Figure 4(b). Letting E'_x and E'_y represent the new set of grid samples, this is written as

$$E'_x(i,j) = \frac{1}{2}\Big[E_x(i,j+1) + E_x(i,j)\Big] \ , \tag{26}$$

$$E'_y(i,j) = \frac{1}{2}\Big[E_y(i+1,j) + E_y(i,j)\Big] \ . \tag{27}$$

Thus, the electric field components are now placed at the same grid locations by defining them along a staggered grid from the voltage potentials. As we shall see in Section 7, such an arrangement also avoids any of the confusion that occurs at the boundaries between dielectric surfaces, since normal field components may be discontinuous here. Finally, the number of rows and columns in the E-field grid are both one less than those of the voltage grid.

6.1 Example 2: The Parallel Plate Capacitor

One of the more interesting simulations readily employed by FDM is the parallel plate capacitor. A demonstration of this scenario is depicted in Figure 5. The simulation domain size was set to 230×135 grid samples, thus giving an over-relaxation constant of $\omega \approx 1.96$. The error bound was

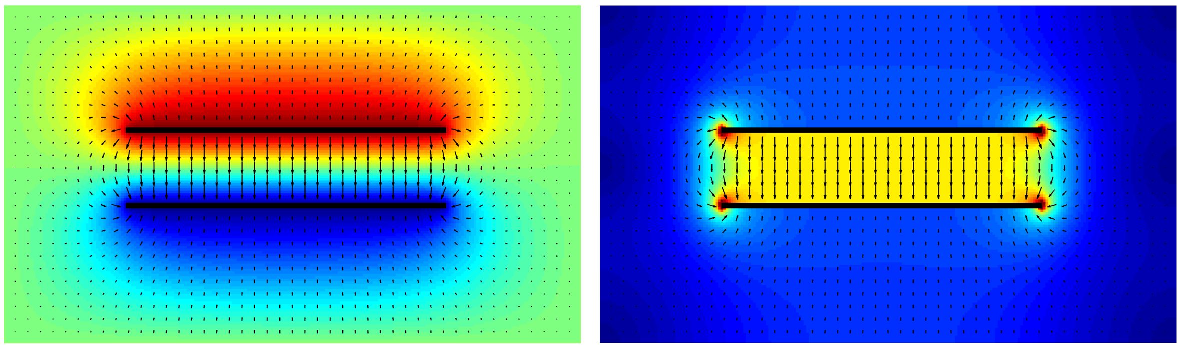

(a) (b)

Figure 5: A 2D parallel-plate capacitor. The top plate is at a potential of +1.0 V while the bottom plate is at −1.0 V. The color mapping represents (a) the voltage potential and (b) the electric field intensity. Quiver plots represent electric field vectors.

set to a value of 10^{-6} and required 427 iterations to converge. The capacitor plates were defined as a set of Dirichlet boundaries fixed at a potential of +1.0 V for the top plate and −1.0 V for the bottom plate,[2] with 0.0 V for the simulation boundaries. The color map in Figure 5(a) represents the voltage potential, with a quiver plot superimposed to represent the electric field vectors. The color map in Figure 5(b) represents the magnitude of the electric field itself. To reduce the effects of the simulation boundaries on the fields near the capacitor, the simulation boundaries were placed many grid points away from the plates as an approximation to a free-space environment.

7 Varying Dielectrics

Let us now return to the generalized Poisson equation:

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] = -\frac{\rho(\mathbf{r})}{\epsilon_0}\ . \tag{28}$$

Although it is tempting to begin by directly applying numerical derivatives to this expression, a far more accurate method may be derived if we first realize that $\epsilon(\mathbf{r})$ does not necessarily need to be sampled at identical grid points as $V(\mathbf{r})$. We shall therefore begin by defining the permittivities as sectionally constant regions along the *staggered* grid shown in Figure 6. Mathematically, this may be written as

$$\epsilon(i,j) = \epsilon(x_i + h/2, y_j + h/2)\ , \tag{29}$$

with $V(i,j)$ and $\rho(i,j)$ defined along the original grid points as before. Defining the permittivities in this way has two important advantages. First, it allows us to define the voltage samples along the boundaries of the dielectric permittivities. This is important when we compute the electric fields from the voltage samples, since electric fields are discontinuous at planar boundaries. The second, and more important reason, is that it allows us to exploit the properties of variational calculus by expressing Equation (28) in its *weak*, or *variational*, form. In so doing, the second-order derivatives vanish and leave only first-order derivatives for us to numerically approximate.

[2]Notice how Dirichlet conditions can also be defined at interior points within the simulation domain and are not limited just to the external boundaries.

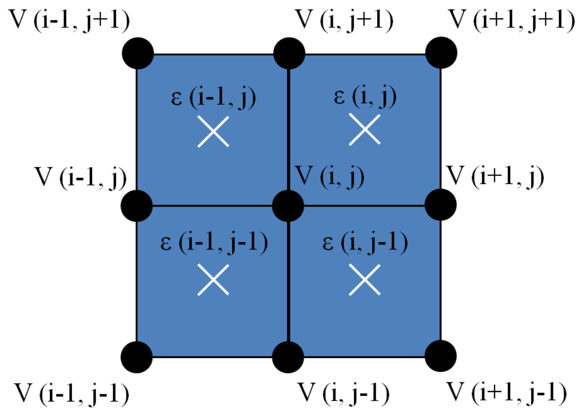


Figure 6: Finite-difference mesh for the generalized Poisson equation. Each blue square represents a region of constant dielectric permittivity.

We begin by defining Ω_{ij} as the square region around a single voltage sample $V(i,j)$, as depicted in Figure 7(a). We then take the surface integral around Ω_{ij} to find

$$\iint\limits_{\Omega_{ij}} \nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big]\, d\Omega = -\frac{1}{\epsilon_0} \iint\limits_{\Omega_{ij}} \rho(\mathbf{r})\, d\Omega\ , \tag{30}$$

where $d\Omega = dxdy$ is the differential surface area. Looking first at the right-hand side of Equation (30), we note that the integral over the charge density is simply the total charge enclosed by Ω_{ij}. We may therefore make the replacement

$$-\iint\limits_{\Omega_{ij}} \rho(\mathbf{r})\, d\Omega = -Q(i,j)\ . \tag{31}$$

The left-hand side of Equation (30) may also be simplified by applying the divergence theorem. This converts the surface integral over Ω_{ij} into a contour integral around its outer border, thus giving

$$\iint\limits_{\Omega_{ij}} \nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big]\, d\Omega = \oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n}\ . \tag{32}$$

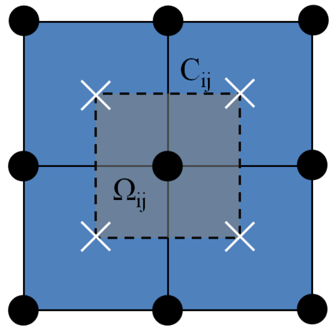
where C_{ij} is the enclosing contour and $d\mathbf{n}$ is the differential unit normal vector. The end result is therefore

$$\oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n} = -\frac{Q(i,j)}{\epsilon_0}\ . \tag{33}$$

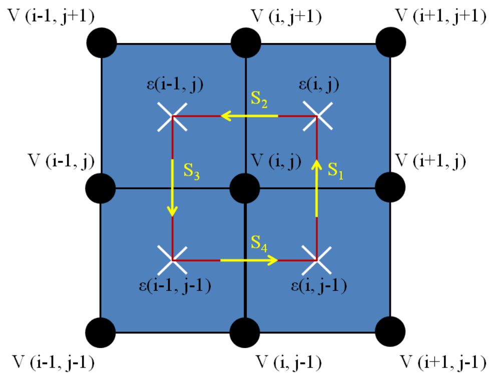
This expression is referred to as the *weak form* of Poisson's equation, while Equation (28) is called the *strong form.*

With the desired expression in hand, the next step is to expand out the gradient operator to find

$$\oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n} = \oint\limits_{C_{ij}} \left[\epsilon(\mathbf{r}) \left(\frac{\partial}{\partial x} V(\mathbf{r})\hat{\mathbf{x}} + \frac{\partial}{\partial y} V(\mathbf{r})\hat{\mathbf{y}}\right)\right] \cdot d\mathbf{n} \tag{34}$$



(a)



(b)

Figure 7: The volume element Ω_{ij} surrounds the voltage sample $V(i,j)$. The outer contour C_{ij} encloses Ω_{ij} and defines the contour integral of Equation (32) in the counter-clockwise direction.

Note that in three dimensions, the surface C_{ij} would normally be a cube, but in our two-dimensional example is simply a square contour. The total contour integral my therefore be broken down by treating it as a series of sub-integrals around each side of the square. For brevity, we shall simply write these integrals as $S_1 \cdots S_4$:

$$\oint_{C_{ij}} \left[\epsilon(\mathbf{r})\left(\frac{\partial}{\partial x}V(\mathbf{r})\hat{\mathbf{x}} + \frac{\partial}{\partial y}V(\mathbf{r})\hat{\mathbf{y}}\right)\right]\cdot d\mathbf{n} = \int_{S_1} + \int_{S_2} + \int_{S_3} + \int_{S_4} \ . \tag{35}$$

This geometry is depicted in Figure 7(b), which highlights the contour of integration over all four sides, taken in the counter-clockwise direction.

To begin, let us define S_1 as the right edge of the square where $d\mathbf{n} = \hat{\mathbf{x}}\,dy$. For convenience, it also helps to assume that $V(i,j)$ lies at the origin, though the end result will be equivalent no matter what location we choose. We may therefore express the integral over S_1 as

$$\int_{S_1} = \int_{-h/2}^{h/2} \epsilon(x,y)\left(\frac{\partial}{\partial x}V(x,y)\,\hat{\mathbf{x}} + \frac{\partial}{\partial y}V(x,y)\,\hat{\mathbf{y}}\right)\cdot\hat{\mathbf{x}}\,dy = \int_{-h/2}^{h/2} \epsilon(x,y)\frac{\partial}{\partial x}V(x,y)\,dy \ . \tag{36}$$

Next, we note that the contour integral is taken entirely along the border between the regions defined by $V(i,j)$ and $V(i+1,j)$. We may therefore approximate the partial derivative by using a central difference between the two samples and then assume it remains constant across the entire border. Calculating the integral across the two dielectric regions therefore gives

$$\begin{aligned}\int_{S_1} &\approx h\left[\frac{\epsilon(i,j)+\epsilon(i,j-1)}{2}\right]\left[\frac{V(i+1,j)-V(i,j)}{h}\right] \\ &= (1/2)\Big[\epsilon(i,j)+\epsilon(i,j-1)\Big]\Big[V(i+1,j)-V(i,j)\Big] \ .\end{aligned} \tag{37}$$

Carrying out this same operation over the other three sides thus gives

$$\int_{S_2} \approx (1/2)\Big[\epsilon(i-1,j)+\epsilon(i,j)\Big]\Big[V(i,j+1)-V(i,j)\Big] \tag{38}$$

$$\int_{S_3} \approx (1/2)\Big[\epsilon(i-1,j-1)+\epsilon(i-1,j)\Big]\Big[V(i-1,j)-V(i,j)\Big] \tag{39}$$

$$\int_{S_4} \approx (1/2)\Big[\epsilon(i,j-1)+\epsilon(i-1,j-1)\Big]\Big[V(i,j-1)-V(i,j)\Big]\ . \tag{40}$$

For notational compactness, we now define the following constants:

$$\begin{aligned}
a_0 &= \epsilon(i,j)+\epsilon(i-1,j)+\epsilon(i,j-1)+\epsilon(i-1,j-1)\\
a_1 &= (1/2)\Big[\epsilon(i,j)+\epsilon(i,j-1)\Big]\\
a_2 &= (1/2)\Big[\epsilon(i-1,j)+\epsilon(i,j)\Big]\\
a_3 &= (1/2)\Big[\epsilon(i-1,j-1)+\epsilon(i-1,j)\Big]\\
a_4 &= (1/2)\Big[\epsilon(i,j-1)+\epsilon(i-1,j-1)\Big]\ .
\end{aligned}$$

Finally, we put it all together to find

$$\oint_{C_{ij}} \approx -a_0V(i,j)+a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)\ . \tag{41}$$

Including the charge term from Equation (33), we finally arrive at

$$\boxed{-a_0V(i,j)+a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)=-\frac{Q(i,j)}{\epsilon_0}}\ . \tag{42}$$

Just like Equation (11), this expression represents a numerical stencil for the generalized Poisson equation. It is therefore a straightforward matter to generate a system of linear equations of the form $\mathbf{Ax}=\mathbf{b}$. However, just like before, $\mathbf{A}$ is a very large and sparse matrix, thereby making direct inversion a difficult option. Fortunately, the same procedure of successive over-relaxation can be applied to iteratively find a solution. Like before, we begin by solving for $V(i,j)$:

$$V(i,j)=\frac{1}{a_0}\Big[a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)+\frac{Q(i,j)}{\epsilon_0}\Big]\ . \tag{43}$$

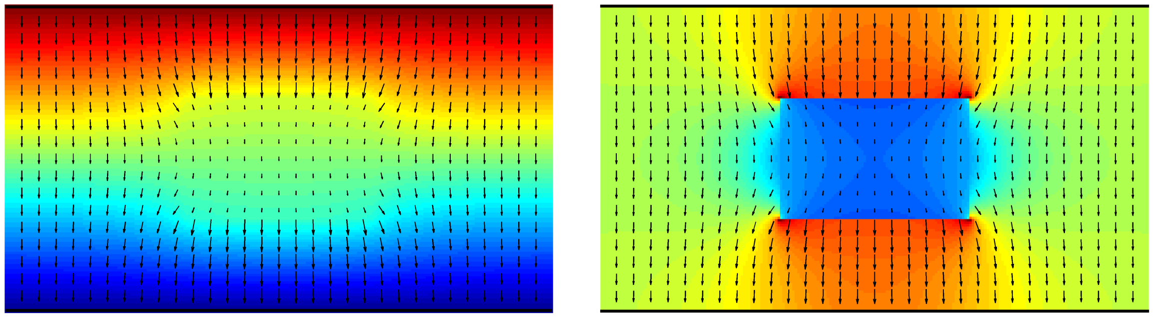
We next define the residual as

$$R(i,j)=\frac{1}{a_0}\Big[a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)+\frac{Q(i,j)}{\epsilon_0}\Big]-V(i,j)\ . \tag{44}$$

And once again, the iteration formula is exactly as we found before:

$$V^{k+1}(i,j)=V^k(i,j)+\omega R^k(i,j)\ . \tag{45}$$

At this point, a word of warning should be emphasized about the use of SOR in regions with multiple dielectrics. Because of the scaling effects of the dielectric constants, SOR tends to converge more slowly as the dielectric constant is increased over some region. As a general rule, the



(a) (b)

Figure 8: A 2D parallel-plate capacitor with a dielectric block ($\epsilon_r = 4$) placed inside. The color maps represent (a) voltage potential and (b) electric field intensity.

convergence rate of SOR will tend to be inversely linear with the maximum dielectric constant in the simulation domain. Thus, for regions with very high dielectrics, the SOR algorithm may even prematurely terminate before converging onto an accurate solution. In these situations, more advanced matrix solvers should be preferred over SOR for both speed and accuracy. However, such topics are beyond the scope of this paper and best left to a more advanced study of inversion theory and numerical methods [3].

7.1 Example 3: Dielectric Block in a Capacitor

Using FDM, simulate a parallel-plate capacitor with a dielectric block placed in between the plates. Use Dirichlet boundaries at the top and bottom with potentials of ± 1.0 V. Use Neumann boundaries at the left and right by setting the normal derivative to zero. Use a dielectric constant of $\epsilon_r = 4$ for the block.

Figure 8 shows an example simulation with the stated parameters. The block is centered between the plates with an aspect ratio of roughly 1.57. The color mapping in Figure 8(a) depicts the voltage potentials while Figure 8(b) depicts the electric field intensity. It is interesting to see how the voltage potential is relatively unperturbed by the presence of the block, while the electric field intensity is dramatically reduced inside the dielectric. The system matrix was inverted by using Matlab's matrix division operator ("\"), which computes quick and accurate inversion of sparse matrices.

8 Quasi-Static Systems

Another useful application for FDM is the ability to solve for quasi-static current density in conductive materials. We begin with Ampere's law in the frequency domain, which is written as

$$\nabla \times \mathbf{H}(\mathbf{r}) = j\omega\epsilon_0\epsilon(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mathbf{J}_c(\mathbf{r}) + \mathbf{J}_i(\mathbf{r}) \ , \tag{46}$$

where $\mathbf{H}$ is the magnetic field intensity, $\mathbf{J}_c$ is the induced conduction current, and $\mathbf{J}_i$ is an impressed current source. At this point, the reader should bear in mind that all functions are complex-valued phasors. We also use $j = \sqrt{-1}$ to denote the imaginary unit, and should not to be confused with an integer index. The conduction current $\mathbf{J}_c$ represents any motion of charges that are due to the flow of electrons on a conductive material in the presence of an external electric field. These are computed from the point form of Ohm's law, which is given as

$$\mathbf{J}_c(\mathbf{r}) = \sigma(\mathbf{r})\,\mathbf{E}(\mathbf{r})\ , \tag{47}$$

where σ is the material conductivity. The $\mathbf{J}_i$ term is an arbitrary mathematical forcing function that represents the flow of electrical currents impressed into the system by external agents. As we shall see shortly, $\mathbf{J}_i$ plays the role of the "source" term that ρ played in the static case.

A common trick in electromagnetics is to merge the displacement current and conduction current of Ampere's law into a single, complex quantity. This is accomplished by defining the *complex permittivity* ϵ_c through the relation

$$\epsilon_c(\mathbf{r}) = \epsilon(\mathbf{r}) + \frac{\sigma(\mathbf{r})}{j\omega\epsilon_0}\ . \tag{48}$$

From here, Ampere's law may now be expressed as

$$\nabla \times \mathbf{H}(\mathbf{r}) = j\omega\epsilon_0\epsilon_c(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mathbf{J}_i(\mathbf{r})\ . \tag{49}$$

If we now take the divergence of Ampere's law, the curl term on the left vanishes, leaving

$$\nabla \cdot \Big[\epsilon_c(\mathbf{r})\ \mathbf{E}(\mathbf{r})\Big] = -\frac{1}{j\omega\epsilon_0}\nabla \cdot \mathbf{J}_i(\mathbf{r})\ . \tag{50}$$

Finally, we apply the *charge continuity equation* by replacing $j\omega\tilde{\rho} = -\nabla \cdot \mathbf{J}_i$ to arrive at

$$\nabla \cdot \Big[\epsilon_c(\mathbf{r})\ \mathbf{E}(\mathbf{r})\Big] = \frac{\tilde{\rho}(\mathbf{r})}{\epsilon_0}\ . \tag{51}$$

Note how this is identical to the form of Equation (2), but with a complex permittivity instead of real-valued. The tilde ($\tilde{}$) over the charge-density term is simply a reminder that ρ is now a complex phasor quantity rather than a real, static value.

Because we are now working with a time-varying system, it is important to realize that the electric field no longer satisfies the simple definition $\mathbf{E} = -\nabla V$. To see why, we must examine Faraday's law, which states

$$\nabla \times \mathbf{E}(\mathbf{r}) = -j\omega\mathbf{B}(\mathbf{r})\ , \tag{52}$$

where $\mathbf{B} = \mu\mathbf{H}$ is the magnetic flux density. We now define a vector field $\mathbf{A}$ called the *magnetic vector potential* that satisfies

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r})\ . \tag{53}$$

Plugging back into Faraday's law therefore gives

$$\nabla \times \Big[\mathbf{E}(\mathbf{r}) + j\omega\mathbf{A}(\mathbf{r})\Big] = 0\ . \tag{54}$$

The significance of this expression comes from an identity in vector calculus, which states that if the curl of some vector field is zero, then that field may be defined as the gradient of some undetermined scalar field V. In other words, V satisfies

$$\mathbf{E}(\mathbf{r}) + j\omega\mathbf{A}(\mathbf{r}) = -\nabla V(\mathbf{r})\ . \tag{55}$$

This is the complete definition for voltage potential, and includes both the effects of an external electric field as well as a time-varying magnetic field. It also means that $\mathbf{E}$ must satisfy

$$\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r}) - j\omega\mathbf{A}(\mathbf{r}) \ . \tag{56}$$

Although it is possible to independently solve for both $\mathbf{A}$ and V to find $\mathbf{E}$, the process is extremely difficult and requires a very complex linear system to couple the two quantities together [4]. To avoid this complication, a far more simplified system may be reached if we impose the *quasi-static* approximation:

$$j\omega\mathbf{A}(\mathbf{r}) \approx 0 \ . \tag{57}$$

In other words, the contribution of $\mathbf{A}$ to the total electric field is negligible at low frequencies, and $\mathbf{E} \approx -\nabla V$. This allows us to rewrite Equation (51) as

$$\boxed{\nabla \cdot \Big[\epsilon_c(\mathbf{r}) \ \nabla V(\mathbf{r})\Big] = -\frac{\tilde{\rho}(\mathbf{r})}{\epsilon_0}} \ . \tag{58}$$

Notice how Equation (58) is simply the generalized Poisson equation again, but with complex numbers now instead of purely real. The same FDM algorithm we have just learned may therefore be used to find low-frequency potentials in a time-varying system. The complex values representing V therefore capture the effects of phase shifts due to material conductivity or any arbitrary offsets impressed within $\tilde{\rho}$. Just like before, large values in the dielectric function do not converge well when using SOR, and so more advanced methods are recommended for inverting the resultant linear system $\mathbf{Ax} = \mathbf{b}$. The only restriction is that the frequency must be low enough such that the magnetic vector potential does not contribute any significant values to the electric field.

8.1 Example 4: Finite Conductors Excited by Constant Voltage

Using the complex-valued FDM, simulate a parallel-plate capacitor with two wires feeding the plates. Excite the plates with a single point of voltage at the tip of each wire. Use a conductivity of $\sigma = 10$ S/m for the metal and a frequency of $f = 1.0$ MHz. Assume the grid spacing is $h = 1.0$ mm. Plot the electric field intensity $\mathbf{E}$ and the conduction current density $\mathbf{J}_c$ of the simulation domain.

The results of this simulation are shown in Figure 9. The excitation voltage of the plates was set by defining a single grid point at the tip of each wire as a Dirichlet boundary. The top plate is excited by a $+1.0$ V point and the bottom plate is excited by a -1.0 V point. Because the output voltage is now a time-varying quantity, it is necessary to choose a point in time (say, $t = 0$) before converting the phasors into an instantaneous, time-domain value. The image in Figure 9(a) clearly shows the typical capacitor plate behavior, with virtually zero electric fields inside the actual metal of the plates. The excitation voltage is therefore nearly constant across both capacitor plates, as we should expect them to behave. The image in Figure 9(b) shows the current density of the same system as calculated from Equation (47). In this case, the single point of voltage excitation becomes apparent, as all the current either enters or exits from these two Dirichlet voltage samples. We can also readily see the behavior of the electrical currents as they flow along the inside of the plates.

9 Static Magnetic Fields

Let us now turn our attention to the problem of static magnetic field intensity. Recall from the previous section that we indirectly defined the magnetic vector potential using

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}) \ , \tag{59}$$

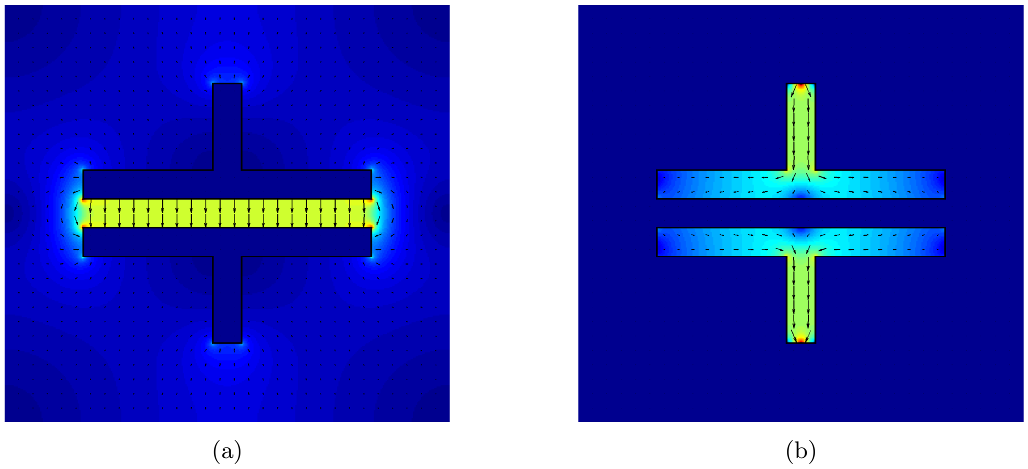


Figure 9: A 2D parallel-plate capacitor with feed-lines. Excitation of the feed-lines was achieved by a single Dirichlet boundary point ($V = \pm 1.0$ V) at the center of each tip. The complex dielectric constant of the metal plates was set to $\epsilon_c = 1 - j1.8 \times 10^5$.

where $\mathbf{B} = \mu \mathbf{H}$. For this problem, we shall even allow for the presence of magnetic materials by defining permeability function as

$$\mu(\mathbf{r}) = \mu_0 \mu_r(\mathbf{r}) \ , \tag{60}$$

where μ_0 is the permeability of free space and μ_r is relative permeability function. Taking the curl of Equation (59) and applying Ampere's law then leads us to the expression

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = j\omega\mu_0\epsilon_0\mu_r(\mathbf{r})\epsilon_c(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r}) \ . \tag{61}$$

For the case of static electric currents, the frequency of excitation is given by $\omega = 0$. We may therefore simplify this expression to read

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = \mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r}) \ . \tag{62}$$

Although the double curl operation may look complex, it is possible to simplify matters by remembering the vector identity

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = \nabla(\nabla \cdot \mathbf{A}(\mathbf{r})) - \nabla^2 \mathbf{A}(\mathbf{r}) \ . \tag{63}$$

It is also important to remember that we have not yet uniquely defined an exact value for $\mathbf{A}$. Instead, all we have done is indirectly define the magnetic vector potential as some unknown vector function such that taking the curl of $\mathbf{A}$ produces $\mathbf{B}$. This does not actually define $\mathbf{A}$ uniquely unless we first provide a *gauge*, or a value set by the divergence $\nabla \cdot \mathbf{A}$. Since we are still technically free to define $\mathbf{A}$ in any way we choose, it helps to choose a gauge that is convenient for a specific problem at hand. In this specific case, the most convenient value is called the *Coulomb gauge*, given by

$$\nabla \cdot \mathbf{A}(\mathbf{r}) = 0 \ . \tag{64}$$

Plugging this back into Equation (62) then produces

$$\boxed{\nabla^2 \mathbf{A}(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r})} \ , \tag{65}$$

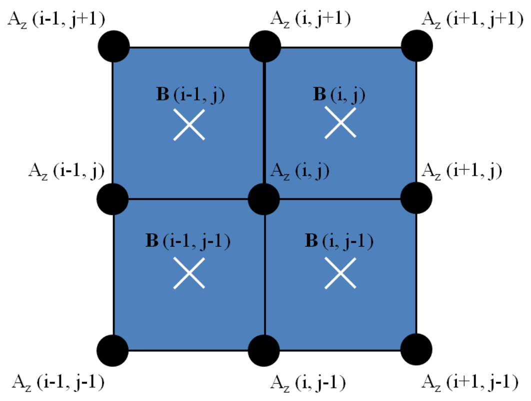


Figure 10: Grid stencil for obtaining the magnetic field samples .

which is nothing more than a vector form of the Poisson equation again. Breaking this up into vector components therefore reveals a set of three independent Poisson equations along each coordinate axis:

$$\nabla^2 A_x(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_x(\mathbf{r}) \tag{66}$$

$$\nabla^2 A_y(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_y(\mathbf{r}) \tag{67}$$

$$\nabla^2 A_z(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_z(\mathbf{r}) \ . \tag{68}$$

The full solution for **A** in three dimensions can therefore be found by independently solving each expression along its respective coordinate axis.

For the special case of two-dimensional systems, we may assume that $A_x = A_y = 0$ and limit ourselves strictly to the special case of electrical currents flowing along the z-direction. This is readily accomplished by once again applying FDM to the Poisson equation and solving for the corresponding values of A_z as if they were voltage potentials. However, magnetic vector potential is more of a mathematical convenience than a useful physical quantity. The quantity we really desire to solve for is the magnetic field intensity **B**. Applying the curl operation from Equation (59) in two dimensions therefore gives us

$$\mathbf{B}(\mathbf{r}) = \hat{\mathbf{x}}\,\frac{\partial A_z}{\partial y} - \hat{\mathbf{y}}\,\frac{\partial A_z}{\partial x} \ . \tag{69}$$

Note that this expression is analogous to the relation between voltage potential and electric field intensity as found in Equation (23). We may therefore define a similar grid stencil between A_z and **B** by using the convention shown in Figure 10.

References

[1] M. N. O. Sakidu, *Numerical Techniques in Electromagnetics*, 2nd ed. Boca Raton, Fl: CRC Press, 2001.

[2] P.-B. Zhou, *Numerical Analysis of Electromagnetic Fields.* Springer Verlag, 1993.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[4] E. Haber, U. M. Ascher, D. A. Aruliah, and D. W. Oldenburg, “Fast simulation of 3D electromagnetic problems using potentials,” *Journal of Computational Physics*, vol. 163, no. 1, pp. 150 – 171, 2000.

Solving the Generalized Poisson Equation Using the Finite-Difference Method (FDM)

James R. Nagel, nageljr@ieee.org
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, Utah

February 15, 2012

1 Introduction

The Poisson equation is a very powerful tool for modeling the behavior of electrostatic systems, but unfortunately may only be solved analytically for very simplified models. Consequently, numerical simulation must be utilized in order to model the behavior of complex geometries with practical value. Although there are several competing algorithms for achieving this goal, one of the simplest and more straightforward of these is called the *finite-difference method* (FDM). At its core, FDM is nothing more than a direct conversion of the Poisson equation from continuous functions and operators into their discretely-sampled counterparts. This converts the entire problem into a system of linear equations that may be readily solved via matrix inversion. The accuracy of such a method is therefore directly tied to the ability of a finite grid to approximate a continuous system, and errors may be arbitrarily reduced by simply increasing the number of samples.

Despite the relative simplicity of FDM as a numerical tool, information on the subject is surprisingly scarce. This is especially true for the case of quasi-static systems experiencing current flow in conducting materials. Part of the reason for this likely has to do with the fact that FDM is, at its core, nothing more than a simplified form of the finite element method (FEM). The only difference is that FDM is solved through a fixed, rectangular geometry, while FEM utilizes a flexible, triangular mesh. Nevertheless, the uniform grids inherent to FDM make it very intuitive to learn and to program, especially for students unfamiliar with techniques in numerical methods. Consequently, the learning curve for FEM is far steeper than it is for FDM, and often requires a whole semester of study to fully understand. On the other hand, expertise with FDM may be readily achieved in only a few weeks, and even serves as an intuitive springboard from which to study the more complex nature of FEM.

The goal of this paper is to serve as a comprehensive introduction to the principles of FDM. Much of the basic information is readily found in standard textbooks [1, 2], though many of the practical details and advanced topics are difficult to find anywhere at all. This paper is therefore a compilation of knowledge based on my experience with FDM, as well as a primer on some of the more advanced topics that are almost nonexistent in the literature. The audience is specifically intended to include first-year students in computational electromagnetics, but more advanced professionals should still find useful reference material as well. The basic governing equations are derived directly from Maxwell's equations and FDM is first introduced in its most basic formulation. The algorithm is then extended from the classical Poisson equation to the generalized Poisson equation in order to include the effects of varying dielectrics within the domain. Finally, we conclude with an extension of FDM to include quasi-static systems by showing how the exact same governing equation still applies for complex-valued phasors.

2 The Generalized Poisson Equation

Beginning with Maxwell's equations, the ultimate governing equation for any electrostatic system is Gauss's law. Expressed in point form, this may be written as

$$\nabla \cdot \mathbf{D}(\mathbf{r}) = \rho(\mathbf{r}) \ . \tag{1}$$

In this context, $\mathbf{r} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}$ is a position vector in space, ρ is the charge density function, and $\mathbf{D}$ is the electric flux density. Using the constitutive relation $\mathbf{D}(\mathbf{r}) = \epsilon_0\epsilon(\mathbf{r})\mathbf{E}(\mathbf{r})$, Gauss's law may be rewritten in terms of the electric field intensity $\mathbf{E}$ as

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \mathbf{E}(\mathbf{r})\Big] = \frac{\rho(\mathbf{r})}{\epsilon_0} \ , \tag{2}$$

where $\epsilon(\mathbf{r})$ is the dielectric constant as a function of position in space and $\epsilon_0 = 8.854 \times 10^{-12}$ F/m is the permittivity of free-space. Gauss's law may be further rewritten in terms of the *voltage potential function* $V(\mathbf{r})$ by making the substitution $\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r})$:

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] = -\frac{\rho(\mathbf{r})}{\epsilon_0} \ . \tag{3}$$

Although it is not commonly discussed in the literature, this is really nothing more than a generalized form of the *Poisson equation*, and is the expression we shall be most interested in throughout this paper. A far more familiar expression occurs if we next assume a uniform dielectric function with the form $\epsilon(\mathbf{r}) = \epsilon_r$. This gives us

$$\nabla^2 V(\mathbf{r}) = -\frac{\rho(\mathbf{r})}{\epsilon_0\epsilon_r} \ , \tag{4}$$

which is the classical form for the Poisson equation as given in most textbooks.

Although the classical Poisson equation is much simpler to numerically solve, it also tends to be very limited in its practical utility. Realistically, the generalized Poisson equation is the true equation we will eventually need to solve if we ever expect to properly model complex physical systems. We shall therefore begin by using the classical Poisson equation as a demonstration case for how FDM works before expanding our algorithm to the generalized form. For brevity and simplicity, this paper will be strictly limited to two-dimensional systems, though a full three-dimensional solution follows a nearly identical derivation.

3 The Five-Point Star

The first step in applying FDM is to define a *mesh*, which is simply a uniform grid of spatial points at which the voltage function will be sampled. Letting h be the distance between each sample, the points that lie on the mesh may be defined by

$$x_i = ih \ , \quad \text{and} \tag{5}$$

$$y_j = jh \ , \tag{6}$$

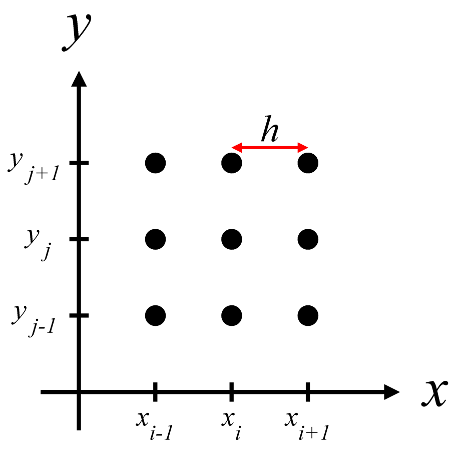


Figure 1: Mesh points for the FDM grid.

where i and j are integers. In practice, i and j will eventually be used as indices for a matrix of voltage samples, so it helps to use a short-hand notation that bears this in mind. We shall therefore replace the spatial coordinates with simple indices by assuming the following convention:

$$V(i,j) = V(x_i, y_j) \ . \tag{7}$$

In a similar fashion, we may also define the charge density samples along the same mesh by using the $\rho(i,j)$ notation.

The next step is to expand the Poisson equation by explicitly showing the partial derivatives in space:

$$\frac{\partial^2 V(i,j)}{\partial x^2} + \frac{\partial^2 V(i,j)}{\partial y^2} = -\frac{\rho(i,j)}{\epsilon_0} \ . \tag{8}$$

The reason for doing this is so that we may approximate the derivative operators through the use of finite-differences. The easiest way to do this is through the three-point approximation for the second-derivative, which is given as

$$\frac{\partial^2}{\partial x^2} V(i,j) \approx \frac{V(i-1,j) - 2V(i,j) + V(i+1,j)}{h^2} \ , \tag{9}$$

with a similar expression for the partial derivative with respect to y. Plugging back into Equation (8) then gives us

$$V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) - 4V(i,j) = -\frac{h^2}{\epsilon_0}\rho(i,j) \ . \tag{10}$$

Finally, we solve for $V(i,j)$ to find

$$V(i,j) = \frac{1}{4}\left[V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) + \frac{\rho(i,j)h^2}{\epsilon_0}\right] \ . \tag{11}$$

What this expression tells us is that every voltage sample $V(i,j)$ is dependent only on $\rho(i,j)$ and the voltage at the four nearest neighbors. A graphical depiction of this is called a *computational molecule*, and is shown in Figure 2. Because of its unique geometry, this five-point stencil is often referred to as the *five point star*.

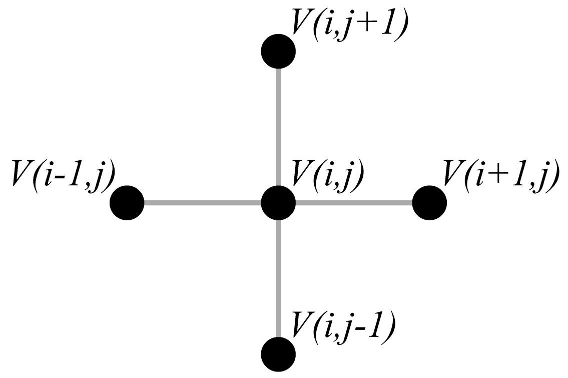


Figure 2: Computational molecule for the 5-point star.

Because each voltage sample $V(i,j)$ is linearly dependent on its four nearest neighbors, the solution over all (i,j) may be represented as a simple matrix-vector equation. This is readily achieved by defining the vector $\mathbf{x}$ to contain all of the voltage samples within the domain. For example, one simple method might scan row-wise along the voltage samples according to the convention:

$$\mathbf{x} = \left[\begin{array}{cccccccc} V(1,1) & V(1,2) & V(1,3) & \cdots & V(2,1) & V(2,2) & V(2,3) & \cdots \end{array}\right]^T . \tag{12}$$

The next step is to express the linear relationship between voltage samples into a matrix $\mathbf{A}$. This effectively converts the entire problem into a matrix-vector equation with the form

$$\mathbf{Ax} = \mathbf{b} , \tag{13}$$

where $\mathbf{b}$ contains all the information about any charge densities and boundary conditions. The numerical solution to the system is finally found by simply inverting the matrix $\mathbf{A}$ to arrive at

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} . \tag{14}$$

4 Boundary Conditions

Because a computer can only store a finite number of grid points, it is always necessary to truncate a simulation domain along some fixed boundary. Since the five-point star is not applicable at the boundary samples, it is necessary to specify boundary conditions in order to arrive at a unique solution to the problem. The two most basic forms of boundary condition are called the *Dirichlet* boundary and the *Neumann boundary.* In practice, it is common for simulations to employ a mixture of these two conditions at the edges, so it is helpful to define Ω_D and Ω_N as the set of all points which satisfy the Dirichlet and Neumann conditions.

The simplest boundary condition is the Dirichlet boundary, which may be written as

$$V(\mathbf{r}) = f(\mathbf{r}) \qquad (\mathbf{r} \in \Omega_D) . \tag{15}$$

The function f is a known set of values that defines V along Ω_D. Thus, the Dirichlet boundary is nothing more than a forced solution to the potential function at specific points. A good example of such a condition occurs in the presence of charged metal plates. Because all points on a metal are

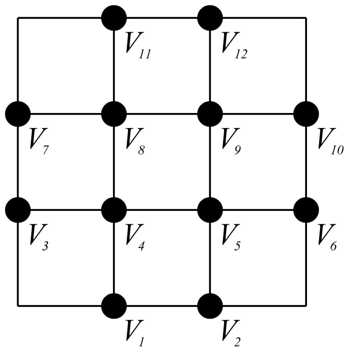


Figure 3: Sampled grid of voltages from Example 1.

at the same potential, a metal plate can readily be modeled by a region of points with some fixed voltage. In contrast, the Neumann boundary condition exists when the derivative of the potential function is known. Generally speaking, the derivative is defined with respect to the outward unit normal at the boundary, which is written as

$$\frac{\partial V(\mathbf{r})}{\partial \mathbf{n}} = f'(\mathbf{r}) \qquad (\mathbf{r} \in \Omega_N) \ , \tag{16}$$

where $\mathbf{n}$ is the outward-pointing unit normal vector, and f' the specifies the set of known derivatives. Unlike the Dirichlet condition, the Neumann does not offer a direct solution to the voltage potentials on a discrete, sampled grid. Rather, the boundary point must be expressed in terms of the surrounding points by applying a new stencil. The simplest method for expressing this is by imagining a central-difference approximation between the boundary sample V_b and the first inner sample V_i:

$$\frac{V_b - V_i}{h} = f' \ . \tag{17}$$

It is important to remember that the derivative function f' is defined with respect to the outward normal direction. This convention allows us to express Equation (17) the same way without any care for where exactly the boundary itself is located, be it top, bottom, left, or right of the simulation domain.

4.1 Example 1: A simple 4 × 4 grid

Consider the simple, 4 × 4 grid of voltage samples depicted in Figure 3. The top boundary is a Dirichlet boundary fixed at 1.0 V with bottom boundary grounded at 0.0 V. The left and right boundaries are Neumann boundaries fixed to a derivative of 0.0 V/m with respect to the outward normal. Using FDM, it is our job to solve for the voltage potentials at all of the indicated points.

The first step is to establish some sort of numbering convention so that the unknown vector $\mathbf{x}$ may be defined. One straightforward way to do this is by scanning across the rows, as indicated by the numbering in Figure 3. It is also worth emphasizing that the samples along the corners of the domain do not make any difference to the final solution of the problem with respect to the interior points. This is because neither the boundary conditions nor the five-point star will depend on what values are placed within the corners. We will therefore neglect these points entirely from the solution set, though in practice it can often be easier to just assign convenient values to them.[1] The vector of unknowns will therefore be written as

$$\mathbf{x} = \begin{bmatrix} V_1 & V_2 & V_3 & \cdots & V_{12} \end{bmatrix}^T .$$

The next step is to fill the system matrix $\mathbf{A}$. We begin by noting that V_1 and V_2 are both Dirichlet boundaries fixed at 0.0 V. The first two rows in $\mathbf{A}$ are therefore nothing but zeros with a one placed at the diagonal element. The same is also true for V_{11} and V_{12} since these are likewise Dirichlet boundaries. The Neumann boundaries are filled in a similar manner, but with a -1 placed on the column corresponding to the interior point. For V_3 and V_7, this is the first element to the right of the diagonal. For V_6 and V_{10}, the -1 is placed at the first element to the left of the diagonal. For the remainder of the samples, the five-point star dictates a value of -4 to be placed at the diagonal, with four 1's placed at their corresponding columns that represent the neighboring points. This will include the two columns immediately adjacent to the diagonal, plus two other 1's placed at the appropriate locations.

The final step is to fill the forcing vector $\mathbf{b}$. Generally speaking, this will be a vector of all zeros except at the points where there is a nonzero boundary condition or a nonzero value for ρ. Thus, $\mathbf{b}$ has only two 1's placed in the last two rows, with zeros placed at all other elements. Writing out the full linear system $\mathbf{Ax} = \mathbf{b}$ therefore leads to

$$\begin{bmatrix}
\mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & \mathbf{1} & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & 0 & \mathbf{1} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{-1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & 0 & 0 & \mathbf{1} & \mathbf{-4} & \mathbf{1} & 0 & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{-1} & \mathbf{1} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1}
\end{bmatrix}
\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ V_{11} \\ V_{12} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix} .$$

Finally, we solve for $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$ to find

$$\mathbf{x} = \begin{bmatrix} 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & \frac{2}{3} & 1 & 1 \end{bmatrix}^T \text{ Volts} .$$

5 Successive Over-Relaxation

For relatively small simulation domains, the direct matrix inversion demonstrated by the previous example works perfectly well for obtaining a solution. However, it is important to realize that the size of $\mathbf{A}$ grows directly with the *square* of the size of $\mathbf{x}$. For example, given a rectangular simulation domain of 100×100 voltage samples, the matrix $\mathbf{A}$ will need to be $10,000 \times 10,000$ elements. Because direct matrix inversion is such an intense operation, it is easy to see how even small simulations can quickly require excessive computational resources.

To reduce the computational cost required by direct matrix inversion, it helps to realize that $\mathbf{A}$ is a *sparse* matrix, meaning the vast majority of elements in $\mathbf{A}$ are all zeros. This is a direct

[1]Imagine a square simulation grid of 200×200 voltage samples. Is it really worth the effort to write a specialized algorithm just to save memory on four measly samples?

consequence of Equation (11), which shows that each voltage element is only dependent on four other samples. As a result, each row in $\mathbf{A}$ has, at most, only five nonzero entries (or even one nonzero entry if the voltage sample is a fixed Dirichlet boundary). This allows us arrive at a solution through the use of sparse matrix solvers that take take advantage of this property. Although there are many available methods to chose from [3], we will focus on a very simple algorithm called *successive over-relaxation* (SOR).

The first step when utilizing SOR is to define the *residual* $R(i,j)$ as the degree to which each voltage sample $V(i,j)$ does not satisfy Equation (11):

$$R(i,j) = \frac{1}{4}\left[V(i-1,j) + V(i+1,j) + V(i,j-1) + V(i,j+1) + \frac{\rho(i,j)h^2}{\epsilon_0}\right] - V(i,j) \tag{18}$$

The next step is to loop over every sample in $V(i,j)$ and add a correction factor defined by $R(i,j)$. This process is then repeated over many iterations until the residual falls below some acceptable error value. For the kth iteration in the loop, we therefore have

$$V^{k+1}(i,j) = V^k(i,j) + R^k(i,j) \ . \tag{19}$$

This method is referred to as *successive relaxation*, and is guaranteed to eventually converge on the correct solution.

Although convergence is a desirable property of successive relaxation, a far more practical property is *rapid* convergence. This may be achieved if we first multiply R with a *relaxation factor* ω, such that

$$V^{k+1}(i,j) = V^k(i,j) + \omega R^k(i,j) \ . \tag{20}$$

This is called the method of successive *over*-relaxation, and will also converge as long as we enforce the condition that $0 < \omega < 2$. The only tricky part is choosing an ideal value for ω that minimizes the time to convergence. Generally speaking, this can only be found through empirical trial-and-error, but a special case arises for rectangular grids. The proof of this is rather involved, so we shall simply state the end result from [1] as

$$\omega = \frac{8 - \sqrt{64 - 16t^2}}{t^2} \ , \tag{21}$$

where N_x and N_y are the number of grid samples in the x- and y-directions, and

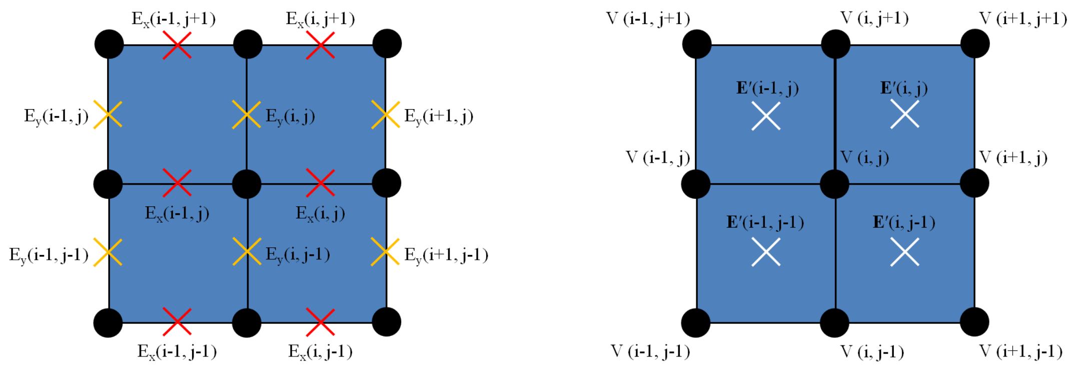
$$t = \cos(\pi/N_x) + \cos(\pi/N_y) \ . \tag{22}$$

6 Electric Fields

From the basic definition $\mathbf{E} = -\nabla V$, it is a straightforward process to extract electric fields from a complex simulation of voltage potentials. We therefore begin by expanding the definition of the gradient into its constituent x and y components:

$$\mathbf{E}(\mathbf{r}) = -\hat{\mathbf{x}}\frac{\partial V(\mathbf{r})}{\partial x} - \hat{\mathbf{y}}\frac{\partial V(\mathbf{r})}{\partial y} \ . \tag{23}$$

The next step is to apply the central difference approximation to the individual field components. The resulting grid of electric field samples is therefore slightly staggered from the voltage potentials



(a) (b)

Figure 4: Grid stencil for obtaining the electric field samples. Circles represent voltage samples while $\times$'s represent the electric field samples. (a) Central differences are first calculated individually along the x and y directions, (b) then averaged together to produce a field sample that is staggered from the voltage grid.

according to the convention shown in Figure 4(a). These components are thus given as

$$E_x(i,j) = -\frac{V(i+1,j) - V(i,j)}{h} \ , \tag{24}$$

$$E_y(i,j) = -\frac{V(i,j+1) - V(i,j)}{h} \ . \tag{25}$$

At this point, two points of information are worth noting. The first is that the E_x grid contains one fewer sample along the x-direction than does the V grid. Similarly, the E_y grid is comprised of one fewer sample along the y-direction. This is simply a natural result of applying the central-difference method to compute a numerical derivative. The second point is that the x- and y-components of the electric field are staggered from each other in space. In order to fix this, we must apply a second approximation by averaging the field components together according to the geometry in Figure 4(b). Letting E'_x and E'_y represent the new set of grid samples, this is written as

$$E'_x(i,j) = \frac{1}{2}\Big[E_x(i,j+1) + E_x(i,j)\Big] \ , \tag{26}$$

$$E'_y(i,j) = \frac{1}{2}\Big[E_y(i+1,j) + E_y(i,j)\Big] \ . \tag{27}$$

Thus, the electric field components are now placed at the same grid locations by defining them along a staggered grid from the voltage potentials. As we shall see in Section 7, such an arrangement also avoids any of the confusion that occurs at the boundaries between dielectric surfaces, since normal field components may be discontinuous here. Finally, the number of rows and columns in the E-field grid are both one less than those of the voltage grid.

6.1 Example 2: The Parallel Plate Capacitor

One of the more interesting simulations readily employed by FDM is the parallel plate capacitor. A demonstration of this scenario is depicted in Figure 5. The simulation domain size was set to 230×135 grid samples, thus giving an over-relaxation constant of $\omega \approx 1.96$. The error bound was

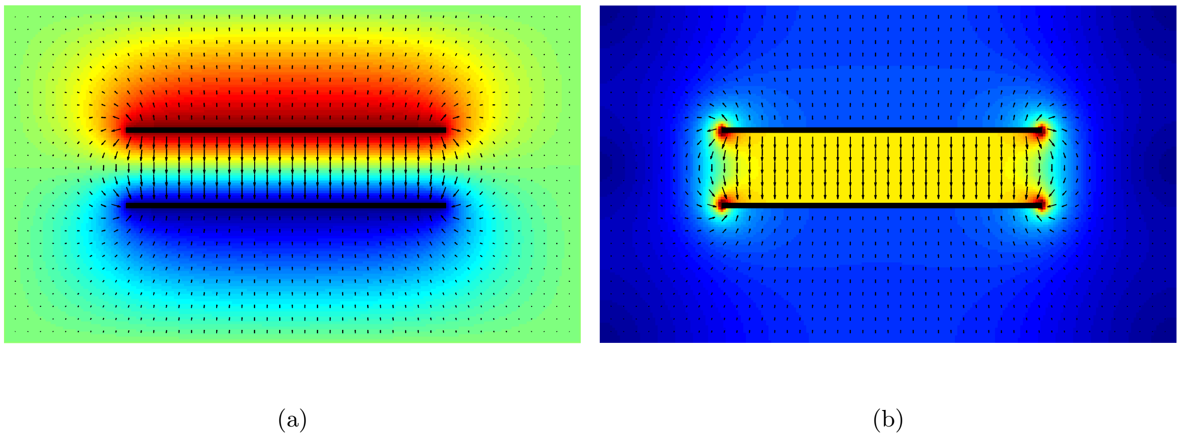


Figure 5: A 2D parallel-plate capacitor. The top plate is at a potential of +1.0 V while the bottom plate is at −1.0 V. The color mapping represents (a) the voltage potential and (b) the electric field intensity. Quiver plots represent electric field vectors.

set to a value of 10^{-6} and required 427 iterations to converge. The capacitor plates were defined as a set of Dirichlet boundaries fixed at a potential of +1.0 V for the top plate and −1.0 V for the bottom plate,[2] with 0.0 V for the simulation boundaries. The color map in Figure 5(a) represents the voltage potential, with a quiver plot superimposed to represent the electric field vectors. The color map in Figure 5(b) represents the magnitude of the electric field itself. To reduce the effects of the simulation boundaries on the fields near the capacitor, the simulation boundaries were placed many grid points away from the plates as an approximation to a free-space environment.

7 Varying Dielectrics

Let us now return to the generalized Poisson equation:

$$\nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] = -\frac{\rho(\mathbf{r})}{\epsilon_0}\ . \tag{28}$$

Although it is tempting to begin by directly applying numerical derivatives to this expression, a far more accurate method may be derived if we first realize that $\epsilon(\mathbf{r})$ does not necessarily need to be sampled at identical grid points as $V(\mathbf{r})$. We shall therefore begin by defining the permittivities as sectionally constant regions along the *staggered* grid shown in Figure 6. Mathematically, this may be written as

$$\epsilon(i,j) = \epsilon(x_i + h/2, y_j + h/2)\ , \tag{29}$$

with $V(i,j)$ and $\rho(i,j)$ defined along the original grid points as before. Defining the permittivities in this way has two important advantages. First, it allows us to define the voltage samples along the boundaries of the dielectric permittivities. This is important when we compute the electric fields from the voltage samples, since electric fields are discontinuous at planar boundaries. The second, and more important reason, is that it allows us to exploit the properties of variational calculus by expressing Equation (28) in its *weak*, or *variational*, form. In so doing, the second-order derivatives vanish and leave only first-order derivatives for us to numerically approximate.

[2]Notice how Dirichlet conditions can also be defined at interior points within the simulation domain and are not limited just to the external boundaries.

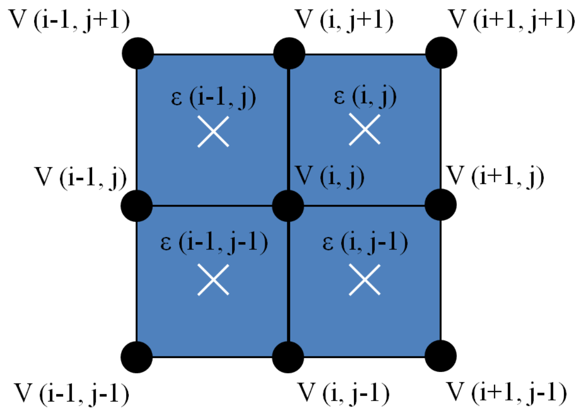


Figure 6: Finite-difference mesh for the generalized Poisson equation. Each blue square represents a region of constant dielectric permittivity.

We begin by defining Ω_{ij} as the square region around a single voltage sample $V(i,j)$, as depicted in Figure 7(a). We then take the surface integral around Ω_{ij} to find

$$\iint\limits_{\Omega_{ij}} \nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big]\, d\Omega = -\frac{1}{\epsilon_0} \iint\limits_{\Omega_{ij}} \rho(\mathbf{r})\, d\Omega\ , \tag{30}$$

where $d\Omega = dxdy$ is the differential surface area. Looking first at the right-hand side of Equation (30), we note that the integral over the charge density is simply the total charge enclosed by Ω_{ij}. We may therefore make the replacement

$$-\iint\limits_{\Omega_{ij}} \rho(\mathbf{r})\, d\Omega = -Q(i,j)\ . \tag{31}$$

The left-hand side of Equation (30) may also be simplified by applying the divergence theorem. This converts the surface integral over Ω_{ij} into a contour integral around its outer border, thus giving

$$\iint\limits_{\Omega_{ij}} \nabla \cdot \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big]\, d\Omega = \oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n}\ . \tag{32}$$

where C_{ij} is the enclosing contour and $d\mathbf{n}$ is the differential unit normal vector. The end result is therefore

$$\oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n} = -\frac{Q(i,j)}{\epsilon_0}\ . \tag{33}$$

This expression is referred to as the *weak form* of Poisson's equation, while Equation (28) is called the *strong form*.

With the desired expression in hand, the next step is to expand out the gradient operator to find

$$\oint\limits_{C_{ij}} \Big[\epsilon(\mathbf{r})\ \nabla V(\mathbf{r})\Big] \cdot d\mathbf{n} = \oint\limits_{C_{ij}} \left[\epsilon(\mathbf{r}) \left(\frac{\partial}{\partial x} V(\mathbf{r})\hat{\mathbf{x}} + \frac{\partial}{\partial y} V(\mathbf{r})\hat{\mathbf{y}}\right)\right] \cdot d\mathbf{n} \tag{34}$$

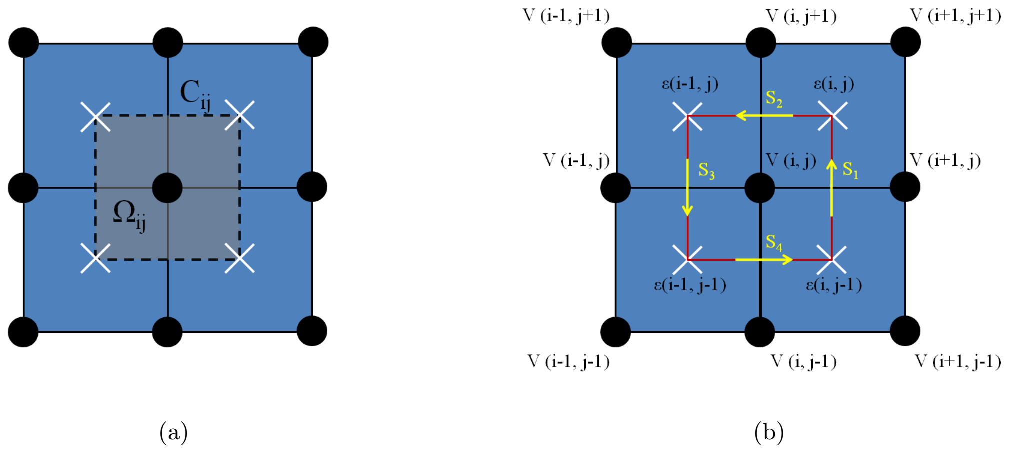


Figure 7: The volume element Ω_{ij} surrounds the voltage sample $V(i,j)$. The outer contour C_{ij} encloses Ω_{ij} and defines the contour integral of Equation (32) in the counter-clockwise direction.

Note that in three dimensions, the surface C_{ij} would normally be a cube, but in our two-dimensional example is simply a square contour. The total contour integral my therefore be broken down by treating it as a series of sub-integrals around each side of the square. For brevity, we shall simply write these integrals as $S_1 \cdots S_4$:

$$\oint_{C_{ij}} \left[\epsilon(\mathbf{r}) \left(\frac{\partial}{\partial x} V(\mathbf{r})\hat{\mathbf{x}} + \frac{\partial}{\partial y} V(\mathbf{r})\hat{\mathbf{y}} \right) \right] \cdot d\mathbf{n} = \int_{S_1} + \int_{S_2} + \int_{S_3} + \int_{S_4} . \tag{35}$$

This geometry is depicted in Figure 7(b), which highlights the contour of integration over all four sides, taken in the counter-clockwise direction.

To begin, let us define S_1 as the right edge of the square where $d\mathbf{n} = \hat{\mathbf{x}}\,dy$. For convenience, it also helps to assume that $V(i,j)$ lies at the origin, though the end result will be equivalent no matter what location we choose. We may therefore express the integral over S_1 as

$$\int_{S_1} = \int_{-h/2}^{h/2} \epsilon(x,y) \left(\frac{\partial}{\partial x} V(x,y)\,\hat{\mathbf{x}} + \frac{\partial}{\partial y} V(x,y)\,\hat{\mathbf{y}} \right) \cdot \hat{\mathbf{x}}\,dy = \int_{-h/2}^{h/2} \epsilon(x,y) \frac{\partial}{\partial x} V(x,y)\,dy . \tag{36}$$

Next, we note that the contour integral is taken entirely along the border between the regions defined by $V(i,j)$ and $V(i+1,j)$. We may therefore approximate the partial derivative by using a central difference between the two samples and then assume it remains constant across the entire border. Calculating the integral across the two dielectric regions therefore gives

$$\begin{aligned} \int_{S_1} &\approx h \left[\frac{\epsilon(i,j) + \epsilon(i,j-1)}{2} \right] \left[\frac{V(i+1,j) - V(i,j)}{h} \right] \\ &= (1/2) \Big[\epsilon(i,j) + \epsilon(i,j-1) \Big] \Big[V(i+1,j) - V(i,j) \Big] . \end{aligned} \tag{37}$$

Carrying out this same operation over the other three sides thus gives

$$\int_{S_2} \approx (1/2)\Big[\epsilon(i-1,j)+\epsilon(i,j)\Big]\Big[V(i,j+1)-V(i,j)\Big] \tag{38}$$

$$\int_{S_3} \approx (1/2)\Big[\epsilon(i-1,j-1)+\epsilon(i-1,j)\Big]\Big[V(i-1,j)-V(i,j)\Big] \tag{39}$$

$$\int_{S_4} \approx (1/2)\Big[\epsilon(i,j-1)+\epsilon(i-1,j-1)\Big]\Big[V(i,j-1)-V(i,j)\Big] \ . \tag{40}$$

For notational compactness, we now define the following constants:

$$\begin{aligned}
a_0 &= \epsilon(i,j)+\epsilon(i-1,j)+\epsilon(i,j-1)+\epsilon(i-1,j-1)\\
a_1 &= (1/2)\Big[\epsilon(i,j)+\epsilon(i,j-1)\Big]\\
a_2 &= (1/2)\Big[\epsilon(i-1,j)+\epsilon(i,j)\Big]\\
a_3 &= (1/2)\Big[\epsilon(i-1,j-1)+\epsilon(i-1,j)\Big]\\
a_4 &= (1/2)\Big[\epsilon(i,j-1)+\epsilon(i-1,j-1)\Big] \ .
\end{aligned}$$

Finally, we put it all together to find

$$\oint_{C_{ij}} \approx -a_0V(i,j)+a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1) \ . \tag{41}$$

Including the charge term from Equation (33), we finally arrive at

$$\boxed{-a_0V(i,j)+a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1) = -\frac{Q(i,j)}{\epsilon_0}} \ . \tag{42}$$

Just like Equation (11), this expression represents a numerical stencil for the generalized Poisson equation. It is therefore a straightforward matter to generate a system of linear equations of the form $\mathbf{Ax}=\mathbf{b}$. However, just like before, $\mathbf{A}$ is a very large and sparse matrix, thereby making direct inversion a difficult option. Fortunately, the same procedure of successive over-relaxation can be applied to iteratively find a solution. Like before, we begin by solving for $V(i,j)$:

$$V(i,j) = \frac{1}{a_0}\Big[a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)+\frac{Q(i,j)}{\epsilon_0}\Big] \ . \tag{43}$$

We next define the residual as

$$R(i,j) = \frac{1}{a_0}\Big[a_1V(i+1,j)+a_2V(i,j+1)+a_3V(i-1,j)+a_4V(i,j-1)+\frac{Q(i,j)}{\epsilon_0}\Big]-V(i,j) \ . \tag{44}$$

And once again, the iteration formula is exactly as we found before:

$$V^{k+1}(i,j) = V^k(i,j)+\omega R^k(i,j) \ . \tag{45}$$

At this point, a word of warning should be emphasized about the use of SOR in regions with multiple dielectrics. Because of the scaling effects of the dielectric constants, SOR tends to converge more slowly as the dielectric constant is increased over some region. As a general rule, the

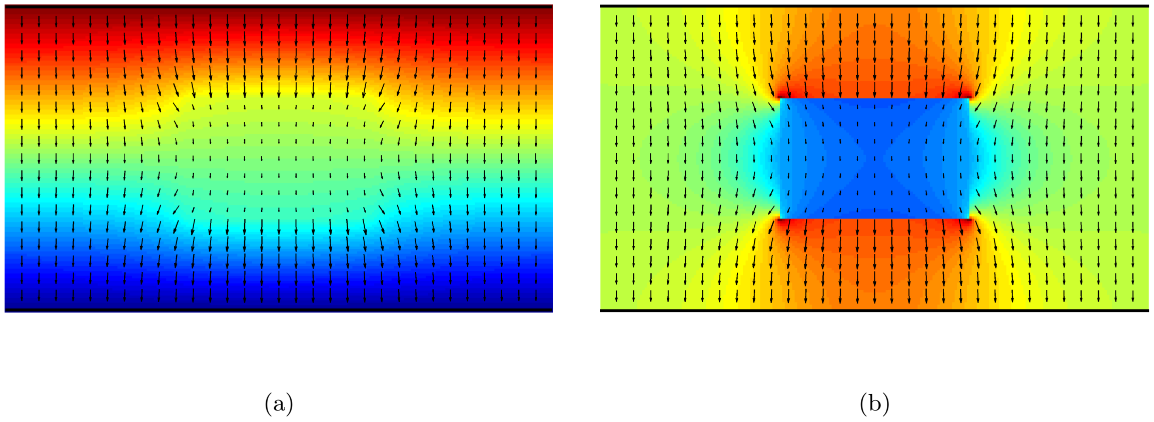

(a) (b)

Figure 8: A 2D parallel-plate capacitor with a dielectric block ($\epsilon_r = 4$) placed inside. The color maps represent (a) voltage potential and (b) electric field intensity.

convergence rate of SOR will tend to be inversely linear with the maximum dielectric constant in the simulation domain. Thus, for regions with very high dielectrics, the SOR algorithm may even prematurely terminate before converging onto an accurate solution. In these situations, more advanced matrix solvers should be preferred over SOR for both speed and accuracy. However, such topics are beyond the scope of this paper and best left to a more advanced study of inversion theory and numerical methods [3].

7.1 Example 3: Dielectric Block in a Capacitor

Using FDM, simulate a parallel-plate capacitor with a dielectric block placed in between the plates. Use Dirichlet boundaries at the top and bottom with potentials of ± 1.0 V. Use Neumann boundaries at the left and right by setting the normal derivative to zero. Use a dielectric constant of $\epsilon_r = 4$ for the block.

Figure 8 shows an example simulation with the stated parameters. The block is centered between the plates with an aspect ratio of roughly 1.57. The color mapping in Figure 8(a) depicts the voltage potentials while Figure 8(b) depicts the electric field intensity. It is interesting to see how the voltage potential is relatively unperturbed by the presence of the block, while the electric field intensity is dramatically reduced inside the dielectric. The system matrix was inverted by using Matlab's matrix division operator ("\"), which computes quick and accurate inversion of sparse matrices.

8 Quasi-Static Systems

Another useful application for FDM is the ability to solve for quasi-static current density in conductive materials. We begin with Ampere's law in the frequency domain, which is written as

$$\nabla \times \mathbf{H}(\mathbf{r}) = j\omega\epsilon_0\epsilon(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mathbf{J}_c(\mathbf{r}) + \mathbf{J}_i(\mathbf{r}) \ , \tag{46}$$

where $\mathbf{H}$ is the magnetic field intensity, $\mathbf{J}_c$ is the induced conduction current, and $\mathbf{J}_i$ is an impressed current source. At this point, the reader should bear in mind that all functions are complex-valued phasors. We also use $j = \sqrt{-1}$ to denote the imaginary unit, and should not to be confused with an integer index. The conduction current $\mathbf{J}_c$ represents any motion of charges that are due to the flow of electrons on a conductive material in the presence of an external electric field. These are computed from the point form of Ohm's law, which is given as

$$\mathbf{J}_c(\mathbf{r}) = \sigma(\mathbf{r})\,\mathbf{E}(\mathbf{r}) \; , \tag{47}$$

where σ is the material conductivity. The $\mathbf{J}_i$ term is an arbitrary mathematical forcing function that represents the flow of electrical currents impressed into the system by external agents. As we shall see shortly, $\mathbf{J}_i$ plays the role of the "source" term that ρ played in the static case.

A common trick in electromagnetics is to merge the displacement current and conduction current of Ampere's law into a single, complex quantity. This is accomplished by defining the *complex permittivity* ϵ_c through the relation

$$\epsilon_c(\mathbf{r}) = \epsilon(\mathbf{r}) + \frac{\sigma(\mathbf{r})}{j\omega\epsilon_0} \; . \tag{48}$$

From here, Ampere's law may now be expressed as

$$\nabla \times \mathbf{H}(\mathbf{r}) = j\omega\epsilon_0\epsilon_c(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mathbf{J}_i(\mathbf{r}) \; . \tag{49}$$

If we now take the divergence of Ampere's law, the curl term on the left vanishes, leaving

$$\nabla \cdot \Big[\epsilon_c(\mathbf{r})\;\mathbf{E}(\mathbf{r})\Big] = -\frac{1}{j\omega\epsilon_0}\nabla \cdot \mathbf{J}_i(\mathbf{r}) \; . \tag{50}$$

Finally, we apply the *charge continuity equation* by replacing $j\omega\tilde{\rho} = -\nabla \cdot \mathbf{J}_i$ to arrive at

$$\nabla \cdot \Big[\epsilon_c(\mathbf{r})\;\mathbf{E}(\mathbf{r})\Big] = \frac{\tilde{\rho}(\mathbf{r})}{\epsilon_0} \; . \tag{51}$$

Note how this is identical to the form of Equation (2), but with a complex permittivity instead of real-valued. The tilde ($\tilde{}$) over the charge-density term is simply a reminder that ρ is now a complex phasor quantity rather than a real, static value.

Because we are now working with a time-varying system, it is important to realize that the electric field no longer satisfies the simple definition $\mathbf{E} = -\nabla V$. To see why, we must examine Faraday's law, which states

$$\nabla \times \mathbf{E}(\mathbf{r}) = -j\omega\mathbf{B}(\mathbf{r}) \; , \tag{52}$$

where $\mathbf{B} = \mu\mathbf{H}$ is the magnetic flux density. We now define a vector field $\mathbf{A}$ called the *magnetic vector potential* that satisfies

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}) \; . \tag{53}$$

Plugging back into Faraday's law therefore gives

$$\nabla \times \Big[\mathbf{E}(\mathbf{r}) + j\omega\mathbf{A}(\mathbf{r})\Big] = 0 \; . \tag{54}$$

The significance of this expression comes from an identity in vector calculus, which states that if the curl of some vector field is zero, then that field may be defined as the gradient of some undetermined scalar field V. In other words, V satisfies

$$\mathbf{E}(\mathbf{r}) + j\omega\mathbf{A}(\mathbf{r}) = -\nabla V(\mathbf{r}) \; . \tag{55}$$

This is the complete definition for voltage potential, and includes both the effects of an external electric field as well as a time-varying magnetic field. It also means that **E** must satisfy

$$\mathbf{E}(\mathbf{r}) = -\nabla V(\mathbf{r}) - j\omega \mathbf{A}(\mathbf{r}) \ . \tag{56}$$

Although it is possible to independently solve for both **A** and V to find **E**, the process is extremely difficult and requires a very complex linear system to couple the two quantities together [4]. To avoid this complication, a far more simplified system may be reached if we impose the *quasi-static* approximation:

$$j\omega \mathbf{A}(\mathbf{r}) \approx 0 \ . \tag{57}$$

In other words, the contribution of **A** to the total electric field is negligible at low frequencies, and $\mathbf{E} \approx -\nabla V$. This allows us to rewrite Equation (51) as

$$\boxed{\nabla \cdot \Big[\epsilon_c(\mathbf{r}) \ \nabla V(\mathbf{r})\Big] = -\frac{\tilde{\rho}(\mathbf{r})}{\epsilon_0}} \ . \tag{58}$$

Notice how Equation (58) is simply the generalized Poisson equation again, but with complex numbers now instead of purely real. The same FDM algorithm we have just learned may therefore be used to find low-frequency potentials in a time-varying system. The complex values representing V therefore capture the effects of phase shifts due to material conductivity or any arbitrary offsets impressed within $\tilde{\rho}$. Just like before, large values in the dielectric function do not converge well when using SOR, and so more advanced methods are recommended for inverting the resultant linear system $\mathbf{Ax} = \mathbf{b}$. The only restriction is that the frequency must be low enough such that the magnetic vector potential does not contribute any significant values to the electric field.

8.1 Example 4: Finite Conductors Excited by Constant Voltage

Using the complex-valued FDM, simulate a parallel-plate capacitor with two wires feeding the plates. Excite the plates with a single point of voltage at the tip of each wire. Use a conductivity of $\sigma = 10$ S/m for the metal and a frequency of $f = 1.0$ MHz. Assume the grid spacing is $h = 1.0$ mm. Plot the electric field intensity **E** and the conduction current density $\mathbf{J}_c$ of the simulation domain.

The results of this simulation are shown in Figure 9. The excitation voltage of the plates was set by defining a single grid point at the tip of each wire as a Dirichlet boundary. The top plate is excited by a $+1.0$ V point and the bottom plate is excited by a -1.0 V point. Because the output voltage is now a time-varying quantity, it is necessary to choose a point in time (say, $t = 0$) before converting the phasors into an instantaneous, time-domain value. The image in Figure 9(a) clearly shows the typical capacitor plate behavior, with virtually zero electric fields inside the actual metal of the plates. The excitation voltage is therefore nearly constant across both capacitor plates, as we should expect them to behave. The image in Figure 9(b) shows the current density of the same system as calculated from Equation (47). In this case, the single point of voltage excitation becomes apparent, as all the current either enters or exits from these two Dirichlet voltage samples. We can also readily see the behavior of the electrical currents as they flow along the inside of the plates.

9 Static Magnetic Fields

Let us now turn our attention to the problem of static magnetic field intensity. Recall from the previous section that we indirectly defined the magnetic vector potential using

$$\mathbf{B}(\mathbf{r}) = \nabla \times \mathbf{A}(\mathbf{r}) \ , \tag{59}$$

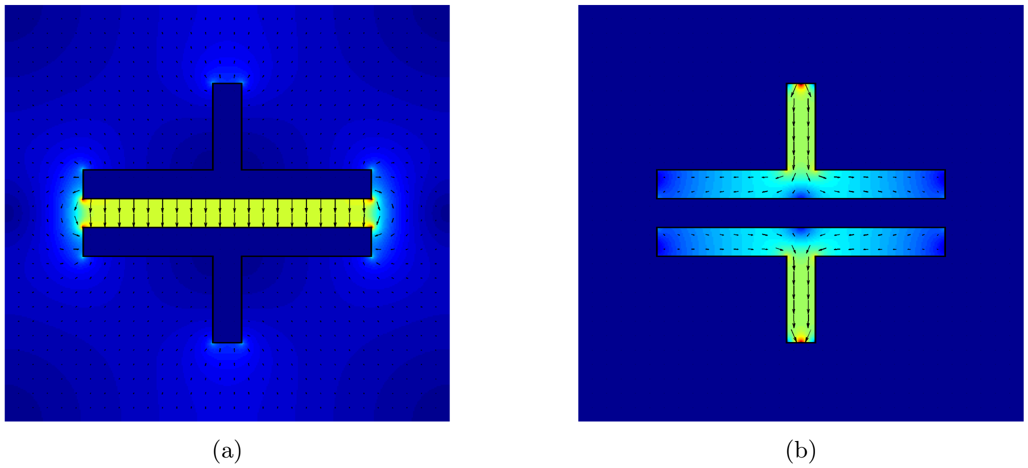


Figure 9: A 2D parallel-plate capacitor with feed-lines. Excitation of the feed-lines was achieved by a single Dirichlet boundary point ($V = \pm 1.0$ V) at the center of each tip. The complex dielectric constant of the metal plates was set to $\epsilon_c = 1 - j1.8 \times 10^5$.

where $\mathbf{B} = \mu \mathbf{H}$. For this problem, we shall even allow for the presence of magnetic materials by defining permeability function as

$$\mu(\mathbf{r}) = \mu_0 \mu_r(\mathbf{r}) \ , \tag{60}$$

where μ_0 is the permeability of free space and μ_r is relative permeability function. Taking the curl of Equation (59) and applying Ampere's law then leads us to the expression

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = j\omega\mu_0\epsilon_0\mu_r(\mathbf{r})\epsilon_c(\mathbf{r})\,\mathbf{E}(\mathbf{r}) + \mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r}) \ . \tag{61}$$

For the case of static electric currents, the frequency of excitation is given by $\omega = 0$. We may therefore simplify this expression to read

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = \mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r}) \ . \tag{62}$$

Although the double curl operation may look complex, it is possible to simplify matters by remembering the vector identity

$$\nabla \times \nabla \times \mathbf{A}(\mathbf{r}) = \nabla(\nabla \cdot \mathbf{A}(\mathbf{r})) - \nabla^2 \mathbf{A}(\mathbf{r}) \ . \tag{63}$$

It is also important to remember that we have not yet uniquely defined an exact value for $\mathbf{A}$. Instead, all we have done is indirectly define the magnetic vector potential as some unknown vector function such that taking the curl of $\mathbf{A}$ produces $\mathbf{B}$. This does not actually define $\mathbf{A}$ uniquely unless we first provide a *gauge*, or a value set by the divergence $\nabla \cdot \mathbf{A}$. Since we are still technically free to define $\mathbf{A}$ in any way we choose, it helps to choose a gauge that is convenient for a specific problem at hand. In this specific case, the most convenient value is called the *Coulomb gauge*, given by

$$\nabla \cdot \mathbf{A}(\mathbf{r}) = 0 \ . \tag{64}$$

Plugging this back into Equation (62) then produces

$$\boxed{\nabla^2 \mathbf{A}(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})\mathbf{J}_i(\mathbf{r})} \ , \tag{65}$$

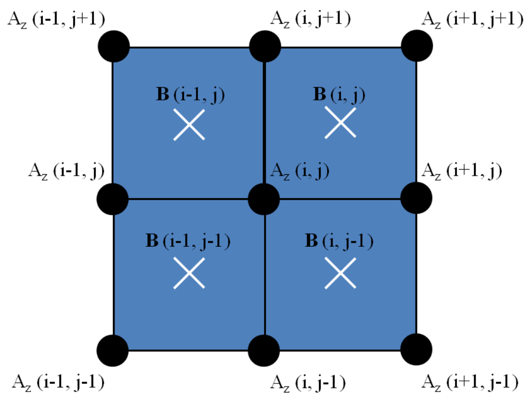


Figure 10: Grid stencil for obtaining the magnetic field samples .

which is nothing more than a vector form of the Poisson equation again. Breaking this up into vector components therefore reveals a set of three independent Poisson equations along each coordinate axis:

$$\nabla^2 A_x(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_x(\mathbf{r}) \tag{66}$$

$$\nabla^2 A_y(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_y(\mathbf{r}) \tag{67}$$

$$\nabla^2 A_z(\mathbf{r}) = -\mu_0\mu_r(\mathbf{r})J_z(\mathbf{r}) \ . \tag{68}$$

The full solution for **A** in three dimensions can therefore be found by independently solving each expression along its respective coordinate axis.

For the special case of two-dimensional systems, we may assume that $A_x = A_y = 0$ and limit ourselves strictly to the special case of electrical currents flowing along the z-direction. This is readily accomplished by once again applying FDM to the Poisson equation and solving for the corresponding values of A_z as if they were voltage potentials. However, magnetic vector potential is more of a mathematical convenience than a useful physical quantity. The quantity we really desire to solve for is the magnetic field intensity **B**. Applying the curl operation from Equation (59) in two dimensions therefore gives us

$$\mathbf{B}(\mathbf{r}) = \hat{\mathbf{x}}\frac{\partial A_z}{\partial y} - \hat{\mathbf{y}}\frac{\partial A_z}{\partial x} \ . \tag{69}$$

Note that this expression is analogous to the relation between voltage potential and electric field intensity as found in Equation (23). We may therefore define a similar grid stencil between A_z and **B** by using the convention shown in Figure 10.

References

[1] M. N. O. Sakidu, *Numerical Techniques in Electromagnetics*, 2nd ed. Boca Raton, Fl: CRC Press, 2001.

[2] P.-B. Zhou, *Numerical Analysis of Electromagnetic Fields.* Springer Verlag, 1993.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007.

[4] E. Haber, U. M. Ascher, D. A. Aruliah, and D. W. Oldenburg, "Fast simulation of 3D electromagnetic problems using potentials," *Journal of Computational Physics*, vol. 163, no. 1, pp. 150 – 171, 2000.

This latter result is easy to understand in many physical situations. For instance, consider an arbitrary electrostatic charge distribution $\rho(\mathbf{x})$. Then

$$\nabla^2\Phi = -\rho/\epsilon_0 \qquad \text{in } \mathbb{R}^3,$$
$$\Phi \to 0 \qquad \text{as } |\mathbf{x}| \to \infty.$$

(We assume here that the charge distribution decays rapidly far from the origin.) Using the integral solution of Poisson's equation, with $V = \mathbb{R}^3$, and setting G to be the fundamental solution in 3D,

$$\Phi(\mathbf{x}_0) = \iiint_{\mathbb{R}^3} \frac{\rho(\mathbf{x})}{4\pi\epsilon_0|\mathbf{x}-\mathbf{x}_0|}\,\mathrm{d}V.$$

We can interpret this physically as the superposition of many infinitesimal charge elements $\rho(\mathbf{x})\,\mathrm{d}V$. Each of these is effectively a point charge, and the potential at $\mathbf{x}_0$ from such a point charge (using the standard formula for the electrostatic potential due to a point charge) is just $\rho(\mathbf{x})\,\mathrm{d}V/4\pi\epsilon_0|\mathbf{x}-\mathbf{x}_0|$. Summing over all such infinitesimal elements gives the above result.

2.8 Numerical Solution of Poisson's Equation

Finite Differences

Applying Taylor's theorem to any smooth function f we obtain

$$f(x+\delta x) = f(x) + \delta x\, f'(x) + \tfrac{1}{2}\delta x^2 f''(x) + \tfrac{1}{6}\delta x^3 f'''(x) + O(\delta x^4),$$
$$f(x-\delta x) = f(x) - \delta x\, f'(x) + \tfrac{1}{2}\delta x^2 f''(x) - \tfrac{1}{6}\delta x^3 f'''(x) + O(\delta x^4).$$

We deduce that

$$f'(x) = \frac{f(x+\delta x)-f(x)}{\delta x} + O(\delta x);$$

hence $\big(f(x+\delta x)-f(x)\big)/\delta x$ is a *first order forward finite difference* approximation to $f'(x)$. (First order because the error term, known as the *truncation error*, is $O(\delta x)$.) Similarly, by subtracting the two Taylor expansions above, we obtain

$$f'(x) = \frac{f(x+\delta x)-f(x-\delta x)}{2\delta x} + O(\delta x^2),$$

giving us a second order *central finite difference.*

The same reasoning allows us to find approximants for higher derivatives: for example, a second order central finite difference for the second derivative which we shall use for Poisson's equation is

$$\boxed{f''(x) \approx \frac{f(x+\delta x)-2f(x)+f(x-\delta x)}{\delta x^2}.}$$

We can use this reasoning to find approximants of as high an order as we like: for instance

$$f'(x) = \frac{-f(x+2\delta x) + 8f(x+\delta x) - 8f(x-\delta x) + f(x-2\delta x)}{12\delta x} + O(\delta x^4).$$

Discretization of Poisson's Equation

Suppose we wish to solve Poisson's equation, $\nabla^2\Phi = \sigma$, in two dimensions in some rectangular domain $a \le x \le b$, $c \le y \le d$. We cover the domain with a regular *grid* (or *mesh*) given by $x_i = a + i\delta x$, $y_j = c + j\delta y$ for $i = 0, \dots, m$, $j = 0, \dots, n$: here $\delta x = (b-a)/m$ and $\delta y = (d-c)/n$ are the grid spacings in the x- and y-directions respectively.

At each grid point (i, j) the exact value of the solution Φ is $\Phi(x_i, y_j)$; we shall find an approximate solution at that grid point which we denote $\Phi_{i,j}$. Using second order central finite differences for both the derivatives $\partial^2\Phi/\partial x^2$ and $\partial^2\Phi/\partial y^2$ we obtain the discretization of Poisson's equation on the grid,

$$\frac{\Phi_{i+1,j} - 2\Phi_{i,j} + \Phi_{i-1,j}}{\delta x^2} + \frac{\Phi_{i,j+1} - 2\Phi_{i,j} + \Phi_{i,j-1}}{\delta y^2} = \sigma(x_i, y_j)$$

at each of the *interior* points $0 < i < m$, $0 < j < n$. In addition, we will have boundary conditions (Dirichlet, Neumann, or a mixture of the two) at $i = 0, m$ and $j = 0, n$.

We therefore have a large number of simultaneous linear equations: at every point of the grid, both interior and on the boundary, there is a corresponding equation, so that we have a total of $(m+1)(n+1)$ equations to solve.

It is usual to take $\delta x = \delta y$, which we shall assume from now on, in which case the discretization at interior points reduces to

$$\boxed{\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} = \sigma(x_i, y_j)\delta x^2.}$$

This can also be denoted using a *stencil* (or *template*) as

What do we do for non-rectangular domains? Suppose we wish to find the steady-state temperature T in an annulus $a \le r \le b$. There are two possible simple approaches. The first is to change coordinates to plane polar coordinates (r, θ), whereupon the grid becomes rectangular: $0 \le \theta \le 2\pi$, with an extra boundary condition that $T(r, 0) = T(r, 2\pi)$. The second approach is to approximate the boundary of the annulus using short line segments in only the x- and y-directions. More advanced techniques can use grids with non-rectangular elements.

Example: suppose we wish to solve Poisson's equation in $0 \le x \le 1$, $0 \le y \le 1$ with $\sigma = 2y$ and boundary conditions $\Phi(x, 0) = 0$, $\Phi(0, y) = 0$, $\Phi(1, y) = y$ (Dirichlet) and $\frac{\partial \Phi}{\partial y}(x, 1) = x^2$ (Neumann). We shall use a grid spacing of $\frac{1}{3}$ in both directions (and will not expect to obtain very accurate results because of the large spacing!).

At each of the four interior points we apply the stencil. On the lower, left and right boundaries we have simply

$$\Phi_{0,0} = \Phi_{1,0} = \Phi_{2,0} = \Phi_{3,0} = 0; \qquad \Phi_{0,1} = \Phi_{0,2} = \Phi_{0,3} = 0;$$

$$\Phi_{3,1} = \tfrac{1}{3}, \quad \Phi_{3,2} = \tfrac{2}{3}, \quad \Phi_{3,3} = 1$$

respectively. On the top boundary we must use a finite difference approximation for $\partial\Phi/\partial y$ to obtain

$$\Phi_{1,3} - \Phi_{1,2} = x_1^2 \delta y = \tfrac{1}{27}, \qquad \Phi_{2,3} - \Phi_{2,2} = x_2^2 \delta y = \tfrac{4}{27}.$$

We can therefore gather together the 16 simultaneous equations representing each point of the grid in a single equation involving a 16×16 matrix as follows:

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -4 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\Phi_{0,0} \\ \Phi_{1,0} \\ \Phi_{2,0} \\ \Phi_{3,0} \\ \Phi_{0,1} \\ \Phi_{1,1} \\ \Phi_{2,1} \\ \Phi_{3,1} \\ \Phi_{0,2} \\ \Phi_{1,2} \\ \Phi_{2,2} \\ \Phi_{3,2} \\ \Phi_{0,3} \\ \Phi_{1,3} \\ \Phi_{2,3} \\ \Phi_{3,3}
\end{pmatrix}
=
\begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{2}{27} \\ \frac{2}{27} \\ \frac{1}{3} \\ 0 \\ \frac{4}{27} \\ \frac{4}{27} \\ \frac{2}{3} \\ 0 \\ \frac{1}{27} \\ \frac{4}{27} \\ 1
\end{pmatrix}$$

In theory, the problem is now solved: we simply use Gaussian elimination to obtain

$$\begin{gathered}
\Phi_{0,3} = 0, \qquad \Phi_{1,3} = \tfrac{1}{9}, \qquad \Phi_{2,3} = \tfrac{4}{9}, \qquad \Phi_{3,3} = 1, \\
\Phi_{0,2} = 0, \qquad \Phi_{1,2} = \tfrac{2}{27}, \qquad \Phi_{2,2} = \tfrac{8}{27}, \qquad \Phi_{3,2} = \tfrac{2}{3}, \\
\Phi_{0,1} = 0, \qquad \Phi_{1,1} = \tfrac{1}{27}, \qquad \Phi_{2,1} = \tfrac{4}{27}, \qquad \Phi_{3,1} = \tfrac{1}{3}, \\
\Phi_{0,0} = \Phi_{1,0} = \Phi_{2,0} = \Phi_{3,0} = 0,
\end{gathered}$$

which is our approximate solution to Poisson's equation.

In fact the exact solution of this problem is $\Phi = x^2 y$, and we see that our approximate solution is quite unexpectedly exact! This is linked to the fact that the exact solution contains only powers of x and y no higher than cubes, and the finite difference formula for second derivatives is exact for such functions.

However, in real applications Gaussian elimination is simply not possible. There are $N = (m+1)(n+1)$ unknowns $\Phi_{i,j}$; the matrix in the problem has N^2 elements, and the number of calculations required to perform a full Gaussian elimination is $O(N^3)$. For even a modest grid size (say 100×100) the number of calculations required would be around 10^{12}, which is unreasonable.

Fortunately, we note that the matrix has a *band diagonal* (or *banded*) structure: beyond a certain distance from the leading diagonal, all entries are zero, so that only a small number of the diagonals in the matrix are non-zero. Because of the widespread need to solve Poisson's and related equations efficiently, many specialised algorithms have been developed for such matrices, taking advantage of the large number of zeros to optimise the elimination process. The number of calculations required by these algorithms is typically only $O(N)$.

2.9 Relaxation Techniques for Poisson's Equation

Another approach to solving the discretized form of Poisson's equation is to use an iterative method. These so-called *relaxation* methods start with some initial guess at the solution (which does not need to be a good guess: it could be simply all zeros!) which is then allowed to move slowly towards the true solution. Note that such methods therefore involve two kinds of errors: those caused by the fact that the finite differences used in the discretization are not exact, and those caused by the fact that the initial guess never quite reaches the true solution.

Relaxation methods are typically a little slower than the matrix methods discussed above, but require significantly less computer memory. This can be an important consideration when high accuracy is required and a small grid spacing is being used.

The Jacobi Method

This is the simplest of the relaxation methods. For Poisson's equation in two dimensions using the standard discretization above the algorithm is as follows:

1. Initialise each $\Phi_{i,j}$ to some initial guess.
2. Apply the boundary conditions.
3. For every interior grid point, calculate the quantity

$$\Phi^*_{i,j} = \tfrac{1}{4}(\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - \sigma(x_i, y_j)\delta x^2).$$

4. For every interior grid point, replace the old approximation $\Phi_{i,j}$ with $\Phi^*_{i,j}$.
5. Repeat from step 2 until the difference between the latest two approximations is smaller than some set tolerance everywhere.

Once the tolerance has been reached, we have that

$$\Phi_{i,j} \approx \tfrac{1}{4}(\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - \sigma(x_i, y_j)\delta x^2),$$

i.e.

$$\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j} \approx \sigma(x_i, y_j)\delta x^2.$$

We therefore have an approximate solution to our original simultaneous equations (which were themselves an approximation to Poisson's equation).

Unfortunately, although each iteration of the Jacobi method is quick and easy, it is very slow to converge, especially for large grids. It is therefore impractical, but it forms the basis for other more useful methods.

The Gauss–Seidel Method

This is very similar to the Jacobi method, except that steps 3 and 4 of the algorithm are combined: as soon as $\Phi^*_{i,j}$ has been calculated for a particular grid point it is *immediately* used to replace $\Phi_{i,j}$. The advantages of this method are that at any time it is only necessary to store the value of $\Phi^*_{i,j}$ at one grid point, rather than at all of them; and the convergence turns out to be somewhat faster (though it is still quite slow).

Successive Over-Relaxation (SOR)

The errors in solutions obtained using either the Jacobi or Gauss–Seidel iterations decrease only slowly, and often in a monotonic manner. We can therefore improve on those methods by over-correcting our solution at each step using a different formula for $\Phi^*_{i,j}$:

$$\Phi^*_{i,j} = (1-\omega)\Phi_{i,j} + \tfrac{1}{4}\omega(\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - \sigma(x_i, y_j)\delta x^2)$$

where ω is the relaxation parameter. The value $\omega = 1$ gives the Gauss–Seidel method again; $\omega < 1$ would produce *under*-relaxation, where we keep a proportion of the old solution; and $\omega > 1$ produces *over*-relaxation where we actually move further away from the old solution than we would using Gauss–Seidel.

The best value to use for ω depends on the particular problem being solved, and may also vary as the iterative process proceeds. However, values in the range 1.2 to 1.4 typically produce good results, and in some cases it is possible to determine an optimal value analytically. The number of iterations required using SOR is significantly less than for either Jacobi or Gauss-Seidel, and for large grids it is often the most practical of all methods of solution.

More advanced methods exist with even better convergence rates, such as the multi-grid method which simultaneously uses a number of different grids with different grid spacings. However, the programming effort required becomes much greater, and so these advanced methods are usually implemented using "black box" routines written by experts in numerical analysis.

2.10 Numerical Solution of the Diffusion Equation

The methods developed for numerical solution of Poisson's equation are easily extended to the diffusion equation

$$\frac{\partial\Phi}{\partial t} = k\nabla^2\Phi.$$

For simplicity we shall consider a rectangular two-dimensional domain $0 \le x \le 1$, $0 \le y \le 1$, with some initial value for $\Phi(x, y, t)$ at $t = 0$.

We first introduce a spatial grid just as for Poisson's equation, with $\delta x = \delta y$, and denote our approximation to $\Phi(x_i, y_j, t)$ by $\Phi_{i,j}(t)$. Using the standard finite difference formulae for the spatial second derivatives we obtain

$$\frac{\mathrm{d}\Phi_{i,j}}{\mathrm{d}t} = k(\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j})/\delta x^2$$

at each interior point. Thus we have $(n-1)^2$ coupled ordinary differential equations, which we can solve using various methods. However, we need to take care to ensure that the method we use leads to a *stable* solution.

The Euler Method

To solve these differential equations we can use a first order forward finite difference in time. Introduce a time-step δt, and let $\Phi_{i,j}^{(k)}$ denote our approximation to $\Phi(x_i, y_j, k\delta t)$. Then using

$$\frac{\Phi_{i,j}^{(k+1)} - \Phi_{i,j}^{(k)}}{\delta t}$$

for the finite difference approximation to $\partial\Phi/\partial t$, we obtain the iterative scheme

$$\boxed{\Phi_{i,j}^{(k+1)} = \Phi_{i,j}^{(k)} + \mu(\Phi_{i+1,j}^{(k)} + \Phi_{i-1,j}^{(k)} + \Phi_{i,j+1}^{(k)} + \Phi_{i,j-1}^{(k)} - 4\Phi_{i,j}^{(k)})}$$

where

$$\boxed{\mu = \frac{k\,\delta t}{\delta x^2}}$$

is the *Courant number.* We can now, starting from the known values of $\Phi_{i,j}^{(0)}$, *step forward* in time to $\Phi_{i,j}^{(1)}$ and so on.

Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2008, Article ID 578723, 11 pages
doi:10.1155/2008/578723

Research Article

Exact and Numerical Solutions of Poisson Equation for Electrostatic Potential Problems

Selçuk Yıldırım

Department of Electrical Education, Firat University, 23119 Elazig, Turkey

Correspondence should be addressed to Selçuk Yıldırım, syildirim@firat.edu.tr

Received 19 September 2007; Accepted 17 March 2008

Recommended by Mohammad Younis

Homotopy perturbation method (HPM) and boundary element method (BEM) for calculating the exact and numerical solutions of Poisson equation with appropriate boundary and initial conditions are presented. Exact solutions of electrostatic potential problems defined by Poisson equation are found using HPM given boundary and initial conditions. The same problems are also solved using the BEM. The cell integration approach is used for solving Poisson equation by BEM. The problem region containing the charge density is subdivided into triangular elements. In addition, this paper presents a numerical comparison with the HPM and BEM.

1. Introduction

It is well known that there are many linear and nonlinear partial equations in various fields of science and engineering. The solution of these equations can be obtained by many different methods. In recent years, the studies of the analytical solutions for the linear or nonlinear evolution equations have captivated the attention of many authors. The numerical and seminumerical/analytic solution of linear or nonlinear, ordinary differential equation or partial differential equation has been extensively studied in the resent years. There are several methods have been developed and used in different problems [1–3]. The homotopy perturbation method is relatively new and useful for obtaining both analytical and numerical approximations of linear or nonlinear differential equations [4–7]. This method yields a very rapid convergence of the solution series. The applications of homotopy perturbation method among scientists received more attention recently [8–10]. In this study, we will first concentrate on analytical solution of Poisson equation, using frequently in electrical engineering, in the form of Taylor series by homotopy perturbation method [11–13].

The boundary element method is a numerical technique to solve boundary value problems represented by linear partial differential equations [14] and has some important advantages. The main advantage of the BEM is that it replaces the original problem with an integral equation defined on the boundary of the solution domain. For the case of a homogeneous partial differential equation, the BEM requires only the discretization on the boundary of the domain [15]. If the simulation domain is free from the electric charge, the governing equation is known as Laplace equation. The BEM computes an approximate solution for the boundary integral formulation of Laplace's equation by discretizing the problem boundary into separate elements, each containing a number of collocation nodes.

The distribution of the electrostatic potential can be determined by solving Poisson equation, if there is charge density in problem domain. In this case, the boundary integral equation obtained from Poisson equation has a domain integral. In the BEM, several methods had been developed for solving this integral. These methods are commonly known as cell integration approach, dual reciprocity method (DRM) and multiple reciprocity method (MRM) [16].

The electric field is related to the charge density by the divergence relationship

$$\begin{gathered} E = \text{electric field}, \\ \nabla E = \frac{\rho}{\varepsilon_0}, \quad \rho = \text{charge density}, \\ \varepsilon_0 = \text{permittivity}, \end{gathered} \tag{1.1}$$

and the electric field is related to the electric potential by a gradient relationship

$$E = -\nabla V. \tag{1.2}$$

Therefore the potential is related to the charge density by Poisson equation:

$$\nabla \cdot \nabla V = \nabla^2 V = \frac{-\rho}{\varepsilon_0}, \quad V = \text{electric potential}. \tag{1.3}$$

2. Theory of the numerical methods

2.1. Homotopy perturbation method

Homotopy perturbation method has been suggested to solve boundary value problems in [17–19]. According to this method, a homotopy with an imbedding parameter $p \in [0,1]$ is constructed and the imbedding parameter is considered as a "small parameter". Here, homotopy perturbation method is used to solve analytic solution of Poisson equation with given boundary conditions.

To illustrate this method, we consider the following nonlinear differential equation:

$$A(u) - f(r) = 0, \quad r \in \Omega, \tag{2.1}$$

with boundary condition

$$B\left(u, \frac{\partial u}{\partial n}\right) = 0, \quad r \in \Gamma, \tag{2.2}$$

where $A(u)$ is written as follows:

$$A(u) = L(u) + N(u). \tag{2.3}$$

A is a general differential operator, B is a boundary operator, $f(r)$ is a known analytical function and Γ is the boundary of the domain Ω. The operator A can be generally divided into two parts L and N, where L is linear operator and N is nonlinear operator. Thus, (2.1) can be rewritten as follows:

$$L(u) + N(u) - f(r) = 0. \tag{2.4}$$

By the homotopy technique [20], we obtain a homotopy $v(r,p) : \Omega \times [0,1] \to \Re$ satisfying

$$H(v,p) = (1-p)\big[L(v) - L(u_0)\big] + p\big[A(v) - f(r)\big] = 0, \quad p \in [0,1],\ r \in \Omega, \tag{2.5}$$

where $p \in [0,1]$ is an embedding parameter and u_0 is an initial approximation of (2.1), which satisfies the boundary conditions. Clearly, from (2.5), we have

$$\begin{aligned} H(v,0) &= L(v) = L(u_0) = 0, \\ H(v,1) &= A(v) = f(r) = 0, \end{aligned} \tag{2.6}$$

the changing process of p from zero to unity is just that of $v(r,p)$ from $u_0(r)$ to $u(r)$. In topology this is called deformation and $L(v) - L(u_0)$, $A(v) - f(r)$ are called homotopic.

We consider v as follows:

$$v = v_0 + pv_1 + p^2v_2 + p^3v_3 + \cdots = \sum_{n=0}^{\infty} p^n v_n. \tag{2.7}$$

According to homotopy perturbation method, an acceptable approximation solution of (2.4) can be explained as a series of the power of p,

$$u = \lim_{p\to 1} v = v_0 + v_1 + v_2 + v_3 + \cdots = \sum_{n=0}^{\infty} v_n. \tag{2.8}$$

Convergence of the series (2.8) is given in [20, 21]. Besides, the same results have been discussed in [22–24].

2.2. Boundary element method

Consider the Poisson equation

$$\nabla^2 u = b_0, \tag{2.9}$$

where b_0 is a known function (for the electrostatic problems, according to Gauss law is $b_0 = -\rho/\varepsilon_0$).

We can develop the boundary element method for the solution of $\nabla^2 u = b_0$ in a two-dimensional domain Ω. We must first form an integral equation from the Poisson equation by using a weighted integral equation and then use the Green-Gauss theorem:

$$\int_\Omega (\nabla^2 u - b_0) w_0 d\Omega = \int_\Gamma \frac{\partial u}{\partial n} w_0 d\Gamma - \int_\Omega \nabla u \cdot \nabla w_0 d\Omega. \tag{2.10}$$

To derive the starting equation for the boundary element method, we use the Green-Gauss theorem again on the second integral. This gives

$$\int_\Omega u(\nabla^2 w_0) d\Omega - \int_\Omega b w_0 d\Omega = \int_\Gamma u \frac{\partial w_0}{\partial n} d\Gamma - \int_\Gamma w_0 \frac{\partial u}{\partial n} d\Gamma, \tag{2.11}$$

and thus, the boundary integral equations are obtained for a domain Ω with boundary Γ, where u potential, $\partial u/\partial n$, is derivative with respect to normal of u and w_0 is the known fundamental solution to Laplace's equation applied at point $\xi(w_0 = -(1/2\pi)\ell n r)$. $r = \sqrt{(\xi - x)^2 + (\eta - y)^2}$ (singular at the point $(\xi, \eta) \in \Omega$).

Then, using the property of the Dirac delta from (2.11),

$$\int_\Omega u(\nabla^2 w_0) d\Omega = -\int_\Omega u \delta(\xi - x, \eta - y) d\Omega = -u(\xi, \eta), \quad (\xi, \eta) \in \Omega, \tag{2.12}$$

that is, the domain integral has been replaced by a point value [25].

Thus, from Poisson equation the boundary integral equation is obtained on the boundary:

$$c(\xi)u(\xi) + \int_\Gamma u \frac{\partial w_0}{\partial n} d\Gamma + \int_\Omega b_0 w_0 d\Omega = \int_\Gamma w_0 \frac{\partial u}{\partial n} d\Gamma, \tag{2.13}$$

where

$$c(\xi) = \begin{cases} 1 & \text{in } \Omega, \\ \dfrac{1}{2} & \text{on } \Gamma. \end{cases} \tag{2.14}$$

The boundary integral equation for the internal points is

$$u(\xi) = \int_\Gamma w_0 \frac{\partial u}{\partial n} d\Gamma - \int_\Gamma u \frac{\partial w_0}{\partial n} d\Gamma - \int_\Omega b_0 w_0 d\Omega. \tag{2.15}$$

2.2.1. Cell integration approach

One of solution of domain integral in the BEM is cell integration approach which is the problem region subdivided to triangular elements as done in the finite element method (Figure 1). Domain integral is solved with respect to relationship between all cells and each boundary node by Gauss quadraturemethod.

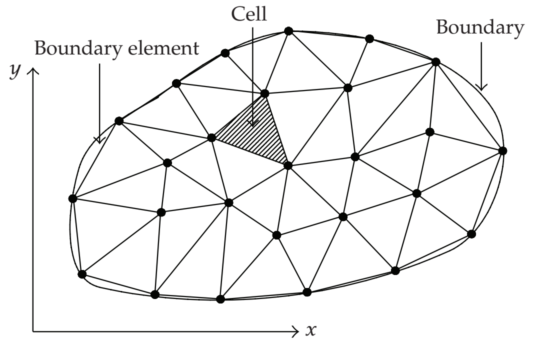


Figure 1: Subdivided regions.

The domain integral in (2.15) for each boundary point i can be written as

$$d_i = \int_{\Omega} b_0 w_0 d\Omega = \sum_{e=1}^{M} \left[\sum_{k=1}^{R} \omega_k \left(b_0 w_0 \right)_k \right] \Omega_e, \tag{2.16}$$

where the integral approximated by a summation over different cells. In (2.16), M is the total number of cells describing the domain Ω, ω_k is the Gauss integration weights and Ω_e is the area of cell e. Besides, the function $(b_0 w_0)$ needs to be evaluated at integration point's k on each cell by 1 to R, see [26].

In this study, a Matlab program has been developed to solve the Poisson equation with BEM by using cell integration approach. This program calculates the potentials in the problem domain.

3. Implementation of homotopy perturbation method to Poisson equation

3.1. Case 1

First, let us investigate exact solution in the y-direction of Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\rho}{\varepsilon_0} = 0, \tag{3.1}$$

with the initial condition

$$u(0, y) = \frac{\rho a^2}{2\varepsilon_0} \left(1 - \frac{32}{\pi^3} \frac{\text{Cosh}(\pi y / 2a)}{\text{Cosh}(\pi b / 2a)} \right), \tag{3.2}$$

and with the Dirichlet boundary conditions (Figure 2); $u = 0$, on $x = \mp 1$ and $y = \mp 1$ (coordinates; $-1 \le x \le 1$ and $-1 \le y \le 1$).

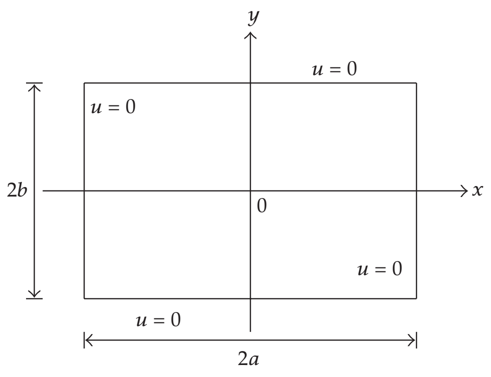


Figure 2: The problem domain and boundary conditions.

To investigate the solution of (3.1), we can construct a homotopy as follows:

$$(1-p)\left[\Upsilon'' - y_0''\right] + p\left[\Upsilon'' + \overset{''}{\Upsilon} + \frac{\rho}{\varepsilon_0}\right] = 0, \tag{3.3}$$

where $\overset{''}{\Upsilon} = \partial^2\Upsilon/\partial y^2$, $\Upsilon'' = \partial^2\Upsilon/\partial x^2$, and $p \in [0,1]$, with initial approximation $\Upsilon_0 = u_0 = (\rho a^2/2\varepsilon_0)(1 - (32/\pi^3)(\text{Cosh}(\pi y/2a)/\text{Cosh}(\pi b/2a)))$. The solution of (3.1) can be expressed in a series in p:

$$\Upsilon = \Upsilon_0 + p\Upsilon_1 + p^2\Upsilon_2 + p^3\Upsilon_3 + \cdots. \tag{3.4}$$

Then, substituting (3.4) into (3.3), and arranging the coefficients of "p" powers, we have

$$\Upsilon_0'' + p\Upsilon_1'' + p^2\Upsilon_2'' + p^3\Upsilon_3'' - y_0'' + py_0'' + p\,\overset{''}{\Upsilon_0} + p^2\,\overset{''}{\Upsilon_1} + p^3\,\overset{''}{\Upsilon_2} + p^4\,\overset{''}{\Upsilon_3} + p\frac{\rho}{\varepsilon_0} + \cdots = 0, \tag{3.5}$$

where the $\Upsilon_i(x,t)$, $i = 1,2,3,\ldots$, are functions to be determined. We have to solve the following system which includes four equations with four unknowns:

$$\begin{aligned}
&p^0:\ \Upsilon_0'' - y_0'' = 0,\\
&p^1:\ \Upsilon_1 = -\iint\left(\frac{\partial^2 y_0}{\partial x^2}\right)dx\,dx - \iint\left(\frac{\partial^2 \Upsilon_0}{\partial y^2}\right)dx\,dx - \iint\left(\frac{\rho}{\varepsilon_0}\right)dx\ dx,\\
&p^2:\ \Upsilon_2 = -\iint\left(\frac{\partial^2 \Upsilon_1}{\partial y^2}\right)dx\,dx,\\
&p^3:\ \Upsilon_3 = -\iint\left(\frac{\partial^2 \Upsilon_2}{\partial y^2}\right)dx\,dx.
\end{aligned} \tag{3.6}$$

To found unknowns $Y_1, Y_2, Y_3, \ldots$, we must use the initial condition (3.2) for the above system, then we obtain

$$\begin{aligned}
Y_0 &= \frac{\rho a^2}{2\varepsilon_0}\left(1 - \frac{32}{\pi^3}\frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)}\right), \\
Y_1 &= \frac{16\rho a^2}{\pi^3 \varepsilon_0}\frac{x^2}{2!}\left(\frac{\pi}{2a}\right)^2 \frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)} - \frac{x^2}{2}\frac{\rho}{\varepsilon_0}, \\
Y_2 &= \frac{16\rho a^2}{\pi^3 \varepsilon_0}\frac{x^4}{4!}\left(\frac{\pi}{2a}\right)^4 \frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)}, \\
Y_3 &= \frac{16\rho a^2}{\pi^3 \varepsilon_0}\frac{x^6}{6!}\left(\frac{\pi}{2a}\right)^6 \frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)}, \\
&\vdots
\end{aligned} \tag{3.7}$$

Thus, as considering (3.4) with (3.7) and using Taylor series, we obtain the analytical solutions as

$$u = \frac{\rho a^2}{2\varepsilon_0} - \frac{16\rho a^2}{\pi^3 \varepsilon_0}\frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)}\left[1 - \frac{1}{2!}\left(\frac{\pi x}{2a}\right)^2 + \frac{1}{4!}\left(\frac{\pi x}{2a}\right)^4 - \frac{1}{6!}\left(\frac{\pi x}{2a}\right)^6 + \cdots\right] - \frac{x^2}{2}\frac{\rho}{\varepsilon_0}. \tag{3.8}$$

Therefore, the exact solution of $u(x, y)$ in closed form is

$$u(x,y) = \frac{\rho a^2}{2\varepsilon_0}\left(1 - \frac{x^2}{a^2}\right) - \frac{16\rho a^2}{\pi^3 \varepsilon_0}\frac{\operatorname{Cosh}(\pi y/2a)}{\operatorname{Cosh}(\pi b/2a)}\operatorname{Cos}\frac{\pi x}{2a}. \tag{3.9}$$

3.2. Case 2

Let us investigate exact solution in the x-direction of Poisson equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\rho}{\varepsilon_0} = 0, \tag{3.10}$$

with the initial condition

$$u(x,0) = \frac{\rho b^2}{2\varepsilon_0}\left(1 - \frac{32}{\pi^3}\frac{\operatorname{Cosh}(\pi x/2b)}{\operatorname{Cosh}(\pi a/2b)}\right), \tag{3.11}$$

and with the Dirichlet boundary conditions. To investigate the solution of (3.10), we can construct a homotopy as follows:

$$(1-p)\left[\overset{''}{Y} - \overset{''}{y_0}\right] + p\left[Y'' + \overset{''}{Y} + \frac{\rho}{\varepsilon_0}\right] = 0. \tag{3.12}$$

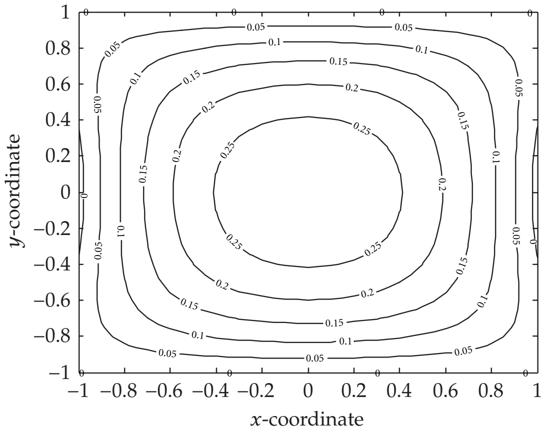


Figure 3: Equipotential lines for $\rho/\varepsilon_0 = 1$ (using HPM, x-direction).

After that, substituting (3.4) into (3.12), and arranging the coefficients of "p" powers, we have to solve the following system including four equations with four unknowns:

$$\begin{aligned}
p^0&:\ Y_0'' - y_0'' = 0,\\
p^1&:\ Y_1 = -\iint\left(\frac{\partial^2 y_0}{\partial x^2}\right)dy\,dy - \iint\left(\frac{\partial^2 Y_0}{\partial y^2}\right)dy\,dy - \iint\left(\frac{\rho}{\varepsilon_0}\right)dy\,dy,\\
p^2&:\ Y_2 = -\iint\left(\frac{\partial^2 Y_1}{\partial x^2}\right)dy\,dy,\\
p^3&:\ Y_3 = -\iint\left(\frac{\partial^2 Y_2}{\partial x^2}\right)dy\,dy.
\end{aligned} \tag{3.13}$$

As found unknowns $Y_1, Y_2, Y_3, \ldots$, we have exact solution of (3.10):

$$u(x,y) = \frac{\rho b^2}{2\varepsilon_0} - \frac{16\,\rho b^2}{\pi^3 \varepsilon_0}\frac{\operatorname{Cosh}(\pi x/2b)}{\operatorname{Cosh}(\pi a/2b)}\left[1 - \frac{1}{2!}\left(\frac{\pi y}{2b}\right)^2 + \frac{1}{4!}\left(\frac{\pi y}{2b}\right)^4 - \frac{1}{6!}\left(\frac{\pi y}{2b}\right)^6 + \cdots\right] - \frac{y^2}{2}\frac{\rho}{\varepsilon_0}. \tag{3.14}$$

Therefore, the exact solution of $u(x,y)$ in closed form is

$$u(x,y) = \frac{\rho b^2}{2\varepsilon_0}\left(1 - \frac{y^2}{b^2}\right) - \frac{16\rho b^2}{\pi^3 \varepsilon_0}\frac{\operatorname{Cosh}(\pi x/2b)}{\operatorname{Cosh}(\pi a/2b)}\operatorname{Cos}\frac{\pi y}{2b}. \tag{3.15}$$

The equipotential lines obtained using exact solution and numerical results have been shown in Figures 3–5 (for $\rho/\varepsilon_0 = 1$). These results then are compared in Tables 1 and 2 (for $\rho/\varepsilon_0 = 1$ and $\rho/\varepsilon_0 = 50$).

Tables 1 and 2 compare the exact HPM and approximate BEM of the Poission equation for $\rho/\varepsilon_0 = 1$ and $\rho/\varepsilon_0 = 50$, respectively. Tables 1 and 2 show that the differences between HPM and BEM for both directions x and y. The differences clearly show that the results of the approximate BEM introduced in this study are acceptable.

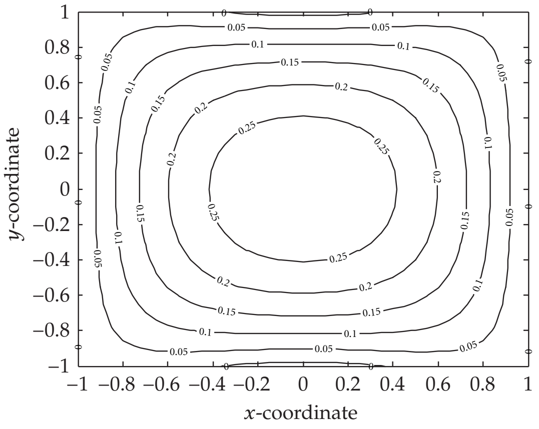


Figure 4: Equipotential lines for $\rho/\varepsilon_0 = 1$ (using HPM, y-direction).

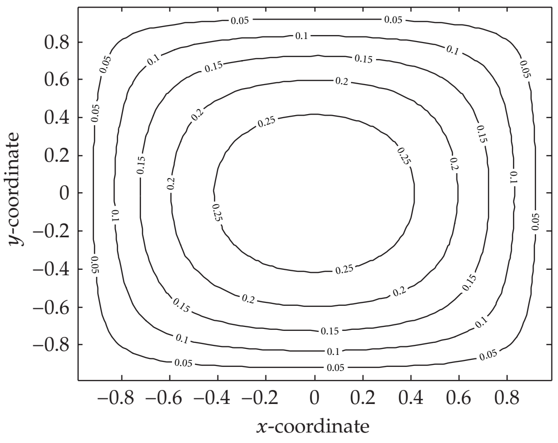


Figure 5: Equipotential lines for $\rho/\varepsilon_0 = 1$ (using BEM).

Table 1: The comparison of potentials (*Volts*) for $\rho/\varepsilon_0 = 1$ at $y = 0$ $(a = 1,\ b = 1)$.

x	HPM (y-direction)	HPM (x-direction)	BEM	|HPM(y)-BEM|	|HPM(x)-BEM|
1.0	0.0	− 0.01602455	0.0	0.0	0.01602455
0.9	0.062828504	0.052243813	0.063177929	0.000349425	0.010934116
0.8	0.116449177	0.109441502	0.116356508	0.000092669	0.006915006
0.7	0.161634683	0.156982714	0.161274602	0.000360081	0.004291888
0.6	0.199119152	0.196042898	0.198740117	0.000379035	0.002697219
0.5	0.229580109	0.227587807	0.229420039	0.000160070	0.001832232
0.4	0.253621787	0.252397383	0.253604246	0.000013624	0.001206863
0.3	0.271760248	0.271085037	0.271809901	0.000049653	0.000724864
0.2	0.284410680	0.284112817	0.284797109	0.000386429	0.000684292
0.1	0.291877170	0.291802833	0.292117395	0.000240225	0.000314562
0.0	0.294345218	0.294345218	0.294569673	0.000224455	0.000224455

Table 2: The comparison of potentials $(Volts)$ for $\rho/\varepsilon_0 = 50$ at $x = 0$ $(a = 1,\ b = 1)$.

y	HPM (y-direction)	HPM (x-direction)	BEM	|HPM(y)-BEM|	|HPM(x)-BEM|
1.0	− 0.80122754	0.0	0.0	0.80122754	0.0
0.9	2.612190681	3.141425217	3.158722460	0.546531779	0.017297243
0.8	5.472075101	5.822458882	5.815100354	0.343025253	0.007358528
0.7	7.849135743	8.081734156	8.083558316	0.234422573	0.001824160
0.6	9.802144924	9.955957630	9.930100041	0.127955117	0.025857589
0.5	11.37939038	11.47900548	11.46598698	0.08659660	0.01301850
0.4	12.61986917	12.68108935	12.67698049	0.05711132	0.00410886
0.3	13.55425187	13.58801241	13.60251771	0.04826584	0.01450530
0.2	14.20564089	14.22053401	14.22555053	0.01990964	0.00501652
0.1	14.59014168	14.59385852	14.62567229	0.03553061	0.03181377
0.0	14.71726094	14.71726094	14.72848367	0.01122273	0.01122273

4. Conclusions

In this paper, we proposed homotopy perturbation method to find exact solution in the x- and y-directions of Poisson equation with appropriate boundary and initial conditions. The numerical results of this electrostatic potential problem have been calculated at the same boundary conditions by BEM. These results are compared with those of HPM in Tables 1 and 2. The obtained numerical results by using BEM are in agreement with the exact solutions obtained by HPM. This adjustment is clearly seen in Figures 3, 4, and 5. It is shown that these methods are acceptable and very efficient for solving electrostatic field problems with charge density.

References

[1] G. Adomian, "A review of the decomposition method in applied mathematics," *Journal of Mathematical Analysis and Applications*, vol. 135, no. 2, pp. 501–544, 1988.

[2] J.-H. He, "Variational iteration method—a kind of non-linear analytical technique: some examples," *International Journal of Non-Linear Mechanics*, vol. 34, no. 4, pp. 699–708, 1999.

[3] S. J. Liao, *Beyond Perturbation: Introduction to Homotopy Analysis Method*, vol. 2 of *CRC Series: Modern Mechanics and Mathematics*, Chapman & Hall/CRC Press, Boca Raton, Fla, USA, 2004.

[4] S. Abbasbandy, "Modified homotopy perturbation method for nonlinear equations and comparison with Adomian decomposition method," *Applied Mathematics and Computation*, vol. 172, no. 1, pp. 431–438, 2006.

[5] J.-H. He, "The homotopy perturbation method nonlinear oscillators with discontinuities," *Applied Mathematics and Computation*, vol. 151, no. 1, pp. 287–292, 2004.

[6] J.-H. He, "Some asymptotic methods for strongly nonlinear equations," *International Journal of Modern Physics B*, vol. 20, no. 10, pp. 1141–1199, 2006.

[7] T. Öziş and A. Yildirim, "A comparative study of He's homotopy perturbation method for determining frequency-amplitude relation of a nonlinear oscillator with discontinuities," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 8, no. 2, pp. 243–248, 2007.

[8] T. Öziş and A. Yildirim, "Traveling wave solution of Korteweg-de Vries equation using He's homotopy perturbation method," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 8, no. 2, pp. 239–242, 2007.

[9] J.-H. He, "New interpretation of homotopy perturbation method," *International Journal of Modern Physics B*, vol. 20, no. 18, pp. 2561–2568, 2006.

[10] S. Abbasbandy, "Application of He' homotopy perturbation method to functional integral equations," *Chaos, Solitons and Fractals*, vol. 31, no. 5, pp. 1243–1247, 2007.

[11] L.-N. Zhang and J.-H. He, "Homotopy perturbation method for the solution of the electrostatic potential differential equation," *Mathematical Problems in Engineering*, vol. 2006, Article ID 83878, 6 pages, 2006.

[12] S. T. Mohyud-Din and M. A. Noor, "Homotopy perturbation method for solving fourth-order boundary value problems," *Mathematical Problems in Engineering*, vol. 2007, Article ID 98602, 15 pages, 2007.

[13] K. Al-Khaled, "Theory and computation in singular boundary value problems," *Chaos, Solitons and Fractals*, vol. 33, no. 2, pp. 678–684, 2007.

[14] C. A. Brebbia and S. Walker, *Boundary Element Techniques in Engineering*, Newnes-Butterworths, London, UK, 1980.

[15] P. K. Kythe, *An Introduction to Boundary Element Method*, CRC Press, Boca Raton, Fla, USA, 1995.

[16] P. W. Partridge, C. A. Brebbia, and L. C. Wrobel, *The Dual Reciprocity Boundary Element Method*, International Series on Computational Engineering, Computational Mechanics and Elsevier Applied Science, Southampton, UK, 1992.

[17] M. A. Noor and S. T. Mohyud-Din, "An efficient algorithm for solving fifth-order boundary value problems," *Mathematical and Computer Modelling*, vol. 45, no. 7-8, pp. 954–964, 2007.

[18] M. A. Noor and S. T. Mohyud-Din, "Homotopy perturbation method for solvingsixth-order boundary value problems," *Computers & Mathematics with Applications*. In press.

[19] M. A. Noor and S. T. Mohyud-Din, "Variational iteration method for solving higher-order nonlinear boundary value problems using He's polynomials," *International Journal of Nonlinear Sciences and Numerical Simulation*, vol. 9, no. 2, 2008.

[20] J.-H. He, "Homotopy perturbation technique," *Computer Methods in Applied Mechanics and Engineering*, vol. 178, no. 3-4, pp. 257–262, 1999.

[21] J.-H. He, "A coupling method of a homotopy technique and a perturbation technique for non-linear problems," *International Journal of Non-Linear Mechanics*, vol. 35, no. 1, pp. 37–43, 2000.

[22] J.-H. He, "Homotopy perturbation method for solving boundary value problems," *Physics Letters A*, vol. 350, no. 1-2, pp. 87–88, 2006.

[23] J.-H. He, "Homotopy perturbation method: a new nonlinear analytical technique," *Applied Mathematics and Computation*, vol. 135, no. 1, pp. 73–79, 2003.

[24] J.-H. He, "Comparison of homotopy perturbation method and homotopy analysis method," *Applied Mathematics and Computation*, vol. 156, no. 2, pp. 527–539, 2004.

[25] P. Hunter and A. Pullan, "FEM/BEM notes," Ph. D. thesis, Department of Engineering Science, University of Auckland, Auckland, New Zealand, 2001.

[26] S. Yıldırım, "The investigation of electric fields in high voltage systems using the boundary element method," Ph. D. thesis, Graduate School of Natural and Applied Science, Firat University, Elazig, Turkey, 1999.