Programming Seminar: 3. Partial Differential Equations: Laplace Equation

Mauricio Sevilla[1,2]

[1]Universidad Nacional de Colombia
[2] E-mail: jmsevillam@unal.edu.co

25th of April of 2018

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Contents

Introduction

Until this point, we've seen some examples of physical systems and the way to solve them numerically.

There are some programs physicists must have done, at least once in a lifetime, this time we'll do one of those, we'll solve the *Laplace equation* using the *Finite difference method*

Partial Differential Equations

On physics we usually find some quantities such as pressure, temperature or electric potential, that are described by *fields* $u(\mathbf{r}, t)$, to find those *fields* we must have a *Partial Differential Equation* to describe them.

Two dimensional case

The most general form for a *Partial Differential Equation* with only two independent variables is as follows:

$$A\frac{\partial^2 u}{\partial x^2} + 2B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} = F \tag{1}$$

Where A, B, C, D, E, and F are arbitrary functions of x and y. We can make a classification of these group of equations using it's characteristic polynomial.

$$Ax^2 + 2Bxy + C^2 + \cdots = 0 \tag{2}$$

We keep the higher orders to study the classification.

Classification

Those polynomials can be classified as *parabolic*, *hyperbolic* and *elliptic*, and that distinction is given by the discriminant of the polynomial such as:

Parabolic	*Hyperbolic*	*Elliptic*
$d = AC - B^2 = 0$	$d = AC - B^2 < 0$	$d = AC - B^2 > 0$
Poisson's	Heat	Wave
$\nabla^2 u(\mathbf{r}) = -4\pi\rho(\mathbf{r})$	$\nabla^2 u(\mathbf{r},t) = a\frac{\partial u(\mathbf{r},t)}{\partial t}$	$\nabla^2 u(\mathbf{r},t) = \frac{1}{v^2}\frac{\partial^2 u(\mathbf{r},t)}{\partial t^2}$

Table 1: Classification of the *Partial Differential Equation* and examples

We usually think about the *hyperbolic* as containing a first-order derivative in one variable and second-order in other, *parabolic* as containing second-order derivatives of all the variables with same signs when placed on the same side, and *elliptic* like the last, but with the different sign.

At any time we solve a *Partial Differential Equation* we must do it with some *Boundary Conditions*.

Let's consider the electrostatic case.

Electrostatics

In electrostatics, the Maxwell equations for the electric field

$$\nabla \cdot \mathbf{E} = 4\pi\rho(\mathbf{r}) \tag{3}$$

$$\nabla \times \mathbf{E} = 0 \tag{4}$$

using that $\nabla \times (\nabla A) = 0$ we can think of the electric field $\mathbf{E}$ as a gradient of a potential function such as:

$$\nabla \cdot \mathbf{E} = \nabla \cdot (-\nabla\varphi) = -\nabla^2\varphi = 4\pi\rho(\mathbf{r}) \tag{5}$$

And that last one, is know as the *Poisson equation*

Now, we want to study the behaviour of this equation, like $\rho(\mathbf{r}) \neq 0$ where there is charge, we have two situations,

- The *Poisson equation*

$$\nabla^2 \varphi = -4\pi\rho(\mathbf{r}) \tag{6}$$

- The *Laplace equation* where there are no charge distributions

$$\nabla^2 \varphi = 0 \tag{7}$$

To solve that numerically, we are going to consider the places with charge as boundary conditions, and solve just the *Laplace equation*.

Finite Difference Method 2D

To implement this method, is necessary divide the space up into a lattice, and solve the equation for each step in the lattice, here we choose a regular lattice in the Cartesian coordinates, the derivation of the solution at each step is made by doing a Taylor expansion as we have done before:

$$\varphi(x+\Delta x, y) = \varphi(x,y) + \frac{\partial \varphi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \varphi}{\partial x^2}(\Delta x)^2 + \cdots \tag{8}$$

$$\varphi(x-\Delta x, y) = \varphi(x,y) - \frac{\partial \varphi}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 \varphi}{\partial x^2}(\Delta x)^2 + \cdots \tag{9}$$

So, if we add these two equations, all the odd terms are cancelled.

so we get an approximation for the second partial derivative as follows :

$$\frac{\partial^2 \varphi(x,y)}{\partial x^2} \approx \frac{\varphi(x+\Delta x, y) + \varphi(x-\Delta x, y) - 2\varphi(x,y)}{(\Delta x)^2} + \mathcal{O}(\Delta x^4) \tag{10}$$

Likewise, for y we have

$$\frac{\partial^2 \varphi(x,y)}{\partial y^2} \approx \frac{\varphi(x, y+\Delta y) + \varphi(x, y-\Delta y) - 2\varphi(x,y)}{(\Delta y)^2} + \mathcal{O}(\Delta y^4) \tag{11}$$

and using that in the Cartesian coordinates the *Poisson equation* goes:

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = -4\pi\rho \tag{12}$$

We just substitute both approximations.

We have an approximate solution as:

$$\frac{\varphi(x+\Delta x,y)+\varphi(x-\Delta x,y)-2\varphi(x,y)}{(\Delta x)^2} + \frac{\varphi(x,y+\Delta y)+\varphi(x,y-\Delta y)-2\varphi(x,y)}{(\Delta y)^2} = -4\pi\rho \quad (13)$$

and if we consider that the lattice in x and y are equal spacings, we have just:

$$\varphi(x+\Delta,y)+\varphi(x-\Delta,y)+\varphi(x,y+\Delta)+\varphi(x,y-\Delta)-4\varphi(x,y) = -4\pi\rho(\Delta)^2 \quad (14)$$

where, we can see a $\varphi(x,y)$ as a function of their neighbours

$$\varphi(x,y) \quad = \quad \frac{1}{4}\left[\varphi(x+\Delta,y)+\varphi(x-\Delta,y)+\ \varphi(x,y+\Delta)+\varphi(x,y-\Delta)\right]\ +\ \pi\rho(\Delta)^2 \quad (15)$$

Due to the discretization of the space, as x as y take some special values, so we can use index instead of coordinates.

$$\varphi_{i,j} = \frac{\varphi_{i+1,j}+\varphi_{i-1,j}+\varphi_{i,j+1}+\varphi_{i,j-1}}{4} + \pi\rho(i\Delta, j\Delta)\Delta^2 \quad (16)$$

in the case where there are no charge distributions:

$$\varphi_{i,j} = \frac{\varphi_{i+1,j}+\varphi_{i-1,j}+\varphi_{i,j+1}+\varphi_{i,j-1}}{4} \quad (17)$$

So, we have that the value of the field φ on the point i,j is given for an average of the values of the field in the next points, as shown in the next figure.

As a schematic representation of the discretization of the space and the value of the field in a point i, j we have

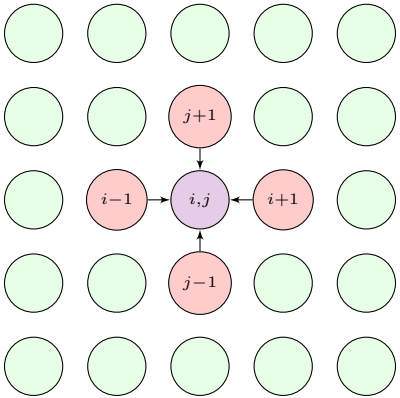

Figure 1: The value of the field φ in the point i, j (The violet) is given by the average of the values of the field in the four nearest neighbours (The red)

Remarks

The equations (16) and (17) as an algoritm, do not provide a direct solution for the *Poisson* and *Laplace* equations, but we must repeat many times until the program converge upon the solution.

This convergence is an important topic, so we may wonder, how many times we have to repeat the method?, can we always get the convergence on a finite time?, and both are up to the system.

Boundaries

Associated to every *Partial Differential Equation*, we have a boundary and initial condition problem, in the case we are studying we have no initial condition problem, just boundary conditions, and we are going to study two of the most important, *Dirichlet* and *Neumann* boundary conditions.

Dirichlet

The *Dirichlet* boundary conditions are the first we are going to use, due to they are easier to implement, and it consist in

$$\varphi(\mathbf{r})|_{\text{Boundary}} = f(\mathbf{r}) \tag{18}$$

where f is a function, and like that condition keeps for every iteration, the positions (i, j) that correspond to the boundaries does not evolve.

Neumann

The case of the *Neumann* boundary conditions has a restriction similar to the *Dirichlet* case, but applies on the normal derivative of the field

$$\left.\frac{\partial \varphi}{\partial \mathbf{n}}\right|_{\text{Boundary}} = \nabla \varphi(\mathbf{r}) \cdot \mathbf{n} = f(\mathbf{r}) \tag{19}$$

where $\mathbf{n}$ is a normal vector of the boundary

Implementation

The implementation of the *Dirichlet* boundary conditions is trivial, so let's think about the *Neumann* problem such as, in the direction of the boundary, let's make an expansion:

$$\varphi(x-\Delta x,y)=\varphi(x,y)-\frac{\partial\varphi(x,y)}{\partial x}\Delta x+\frac{1}{2}\frac{\partial^2\varphi}{\partial x^2}(\Delta x)^2+\cdots \tag{20}$$

$$\varphi(x-2\Delta x,y)=\varphi(x,y)-\frac{\partial\varphi(x,y)}{\partial x}(2\Delta x)+\frac{1}{2}\frac{\partial^2\varphi}{\partial x^2}(2\Delta x)^2+\cdots \tag{21}$$

if we take 4 times the equation (20), we and subtract the last one:

$$\varphi(x,y)\approx\frac{4\varphi(x-\Delta x,y)-\varphi(x-2\Delta x,y)}{3}+\frac{2}{3}\frac{\partial\varphi(x,y)}{\partial x}\Delta x+\mathcal{O}(\Delta x^3) \tag{22}$$

The representation of the situation is:

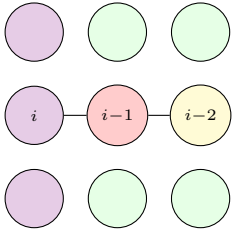

Figure 2: The value of the field φ in the boundary for *Neumann* boundary conditions

So, in the representation of the components for the boundaries we get:

$$\varphi_{i,j} = \frac{4\varphi_{i-1,j} - \varphi_{i-2,j}}{3} + \frac{2}{3}\left(\frac{\partial \varphi_{i,j}}{\partial x}\Delta x + \frac{\partial \varphi_{i,j}}{\partial y}\Delta y\right) \tag{23}$$

Considerations

To do the numerical calculations, we must take a finite region on space where the geometry is placed

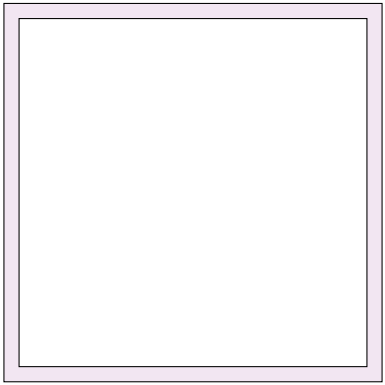

Figure 3: boundary conditions

Considerations

To do the numerical calculations, we must take a finite region on space where the geometry is placed

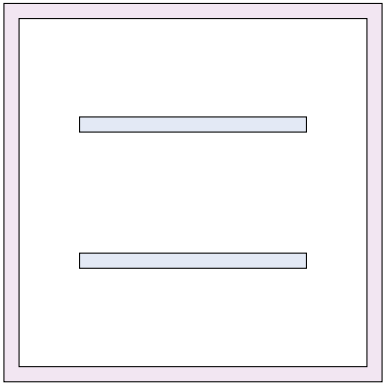

Figure 3: boundary conditions

Considerations

To do the numerical calculations, we must take a finite region on space where the geometry is placed

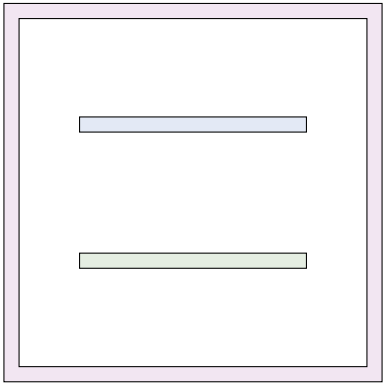

Figure 3: boundary conditions

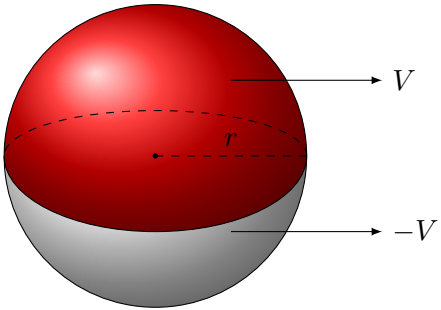

Figure 4: Sphere at two potentials

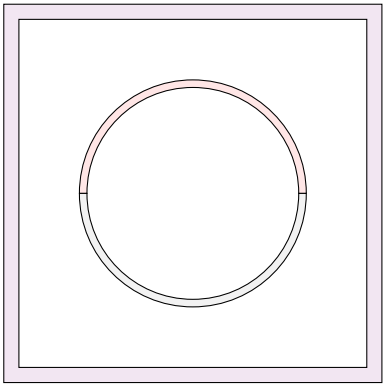

Figure 5: 2D representation[a]

[a] In fact, this is the 2D representation of a infinite cylinder.

Implementation

To do the implementation we'll use a different data type very different than lists.

We're going to use `numpy` arrays.

To define a `numpy` array the list structure is crucial

```
>>> a=np.array([1,2,3])
```

between the parenthesis there is a list.

```
>>> b=[1,2,3]
>>> a=np.array(b)
```

Test features.

How to do it

To do the code, we must do the following steps

- Create a *Matrix* where the information is saved.
- initialize the matrix.
- Calculate the next *time*step.
- Calculate the boundary conditions.

Repeat last two until stabilization.

References I

[1] Mauricio Sevilla.
Programming seminar: 2018-01.
https://github.com/jmsevillam/Seminar, 2018.
[Online; accessed 25-June-2018].

[2] Python Software Foundation.
The python tutorial.
https://docs.python.org/3/tutorial/index.html, 2018.
[Online; accessed 25-June-2018].

[3] NumPy developers.
Quickstart tutorial.
https://docs.scipy.org/doc/numpy/user/quickstart.html, 2018.
[Online; accessed 25-June-2018].

References II

[4] cplusplus.com.
C++ language.
http://www.cplusplus.com/doc/tutorial/, 2018.
[Online; accessed 25-June-2018].

[5] J.D. Jackson.
Classical Electrodynamics.
Wiley, 2012.

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