Numerical Solution to an Electric Potential Problem Using Mathematica's NDSolve Function

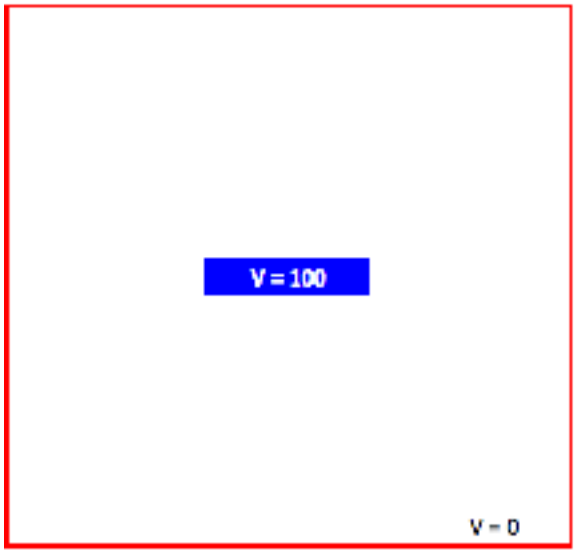
Adapted from https://mathematica.stackexchange.com/questions/59441/solve-laplace-equation-using-ndsolve
(The solution shown below is based on one by Sjoerd C. deVries.

PROBLEM

Consider a solid rectangular rod electrode of infinite length oriented along the z-axis. A second hollow electrode (with square cross-section), also of infinite length symmetrically surrounds the solid electrode and is oriented along the z-axis.

The inner electrode is at a potential V = 100 and the outer electrode is grounded.

This figure shows the 2D representation of the problem; obviously it is a 2D problem because infinite lengths rule out any z-dependence in the potential. Note: Cartesian Coordinates are the natural coordinate system to use here.



(a) First, find numerically the electric potential in the space between the electrodes. I will guide you through this exercise.

The BCs as given are called Fixed BCs or Dirichlet BCs. NDSolve can take BCs as an argument; these are inserted via M's DirichletCondition function. First we define a region called Ω, where we want to find the potential. I will assume the outer boundary of Ω is 100 units on a side. M defines rectangles by the position (x and y coordinates) of the lower left

corner and the position of the upper right corner. Here Rectangle[{0,0}, {100,100}]. The inner boundary of Ω is also a rectangle. I chose the inner boundary to be Rectangle[{40,48}, {60,52}]. We want the potential *between* the rectangles. This region is specified by the RegionDifference[] command—the big rectangle minus the small one.

```
(* Study and execute the code below *)

ClearAll["`*"]
Ω = RegionDifference[Rectangle[{0, 0}, {100, 100}], Rectangle[{40, 48}, {60, 52}]];
sol = NDSolve[{D[V[x, y], x, x] + D[V[x, y], y, y] == 0,
    DirichletCondition[V[x, y] == 100., x == 40 && 48 ≤ y ≤ 52 ||
       x == 60 && 48 ≤ y ≤ 52 || 40 ≤ x ≤ 60 && y == 48 || 40 ≤ x ≤ 60 && y == 52],
    V[x, 0] == V[x, 100] == V[0, y] == V[100, y] == 0}, V, {x, y} ∈ Ω]
```

The output of M's Numerical Solving routines is a construct that you can treat like a function. To look at Wolfram's description run this cell; click on the >> at the end to get the full Monty.

```
(* For information on InterpolatingFunctions, execute the code below *)

? InterpolatingFunction
```

After extracting our InterprelatingFunction (I call it VV[x,y]), we can ask for it's value anywhere it is defined. Note what happens when we go outside Ω.

```
(* Execute code below *)

VV[x_, y_] = V[x, y] /. sol;
(* This code extracts the solution from the InterpolatingFunction*)
VV[25, 25]
VV[300, 300]   (*  outside Ω *)
```

(b) Generate a contour plot of the potential using ContourPlot. (Be brave; remember, VV is an InterprelatingFunction.)

```
(* Input code below *)
```

(c) Comment on your plot.

<Enter comments in this text cell>

(d) Again bravely, apply –Grad[VV[x,y],{x,y}] to find a representation of the electric field in the {x,y} plane. (You end up with TWO InterprelatingFunctions, one for Ex and one for Ey).

It helps to mask the center electrode with a suitably colored rectangle of the appropriate size. This can be created with the Graphics[] command. If you give it a name (I used "pinky", you can superimpose it on your field plot with the Show[] command. For pinky:

pinky = Graphics[{Pink, Rectangle[{40,48},{60,52}];

Plot this E field using VectorPlot (Brave! Be Brave!)

```
(* Input code below *)
```

(e) Generate a StreamPlot of this field.

```
(* Input code below *)
```

(e) Use Show, to superimpose this StreamPlot of E onto the ContourPlot of V. If you add pinky to the list of objects to show, the inner rectangle will appear here too.

```
(* Input code below *)
```

(f) Handshake. State the obvious features if this composite plot.

<Enter your observations in this text cell>