# Weeks 5 & 6: Boundary Value Problems and Finite Differences - Take 2

No matter how much we would like the world to only be in one-dimension (to make our analytical math easier), the world is three-dimensional. Even more, not all engineering processes are steady-state but rather change with time. Worse again, not all boundary value problems (BVPs) using finite differences (FDs)are linear, many result in a system of non-linear equations. Even worse still, not all geometries are rectangular prisms, cylinders, or spheres, making the standard coordinate systems with standard meshing schemes difficult to implement. No matter the additional challenge, we must be able to tackle the general BVP no matter how complicated the resulting implementation may be. Thus, we must move beyond our 1D world and be able to examine boundary value problems in 2D and even 3D.

### Multidimensional boundary value problems

Consider a steady-state 2D BVP on the rectangular domain shown below and following the same PDE we defined last week:

$$0 = -\boldsymbol{\nabla} \cdot (\varphi \mathbf{v}) + \Gamma \boldsymbol{\nabla}^2 \varphi + s\left(t, \mathbf{r}, \varphi\right) \tag{1}$$

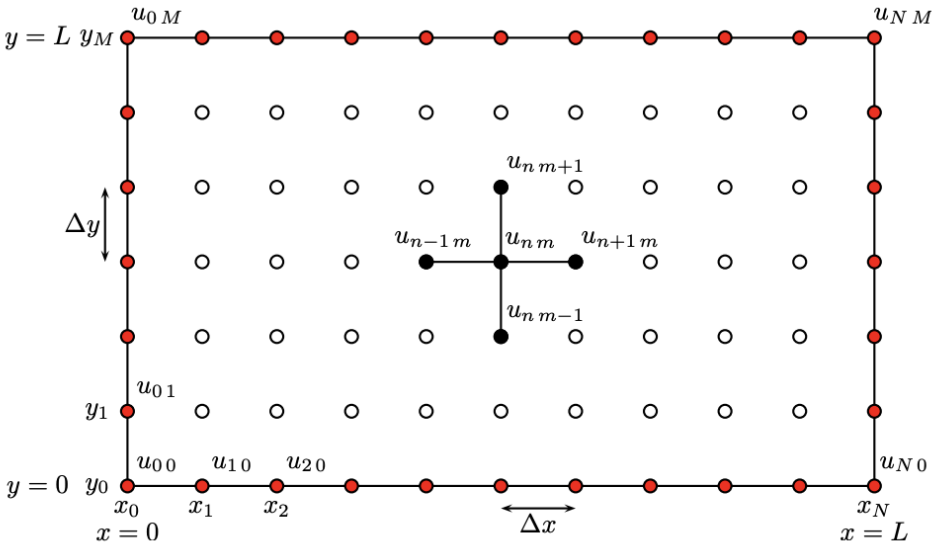



Figure 1: Regular 2D grid

However, this system will have no convection but does include diffusion and source terms, giving the following in 2D:

$$-\boldsymbol{\nabla}^2 \varphi = -\frac{\partial^2 \varphi}{\partial x^2} - \frac{\partial^2 \varphi}{\partial y^2} = s\left(x, y\right) \tag{2}$$

for $0 \leq x \leq L$, $0 \leq y \leq H$, $s\left(x, y\right)$ being a generic function we will define later, and the following boundary conditions:

$$\begin{array}{lll} \text{BC1:} & \varphi(0,y)=0 & 0\le y\le H \\ \text{BC2:} & \varphi(L,y)=0 & 0\le y\le H \\ \text{BC3:} & \varphi(x,0)=0 & 0\le x\le L \\ \text{BC4:} & \varphi(x,H)=0 & 0\le x\le L \end{array}$$

This PDE system is solved by approximating the partial derivatives with the central difference formula:

$$\left(\frac{\partial^2 f}{\partial x^2}\right)\Big|_{x_i,y_j} \approx \frac{f(x_{i+1},y_j)-2f(x_i,y_j)+f\left(x_{i-1,y_j}\right)}{(\Delta x)^2} \tag{3}$$

and

$$\left(\frac{\partial^2 f}{\partial y^2}\right)\Big|_{x_i,y_j} \approx \frac{f(x_i,y_{j+1})-2f(x_i,y_j)+f(x_i,y_{j-1})}{(\Delta y)^2} \tag{4}$$

Substituting (3) and (4) into the PDE we are solving (2) gives the following expression:

$$-\frac{\varphi(x_{i-1},y_j)-2\varphi(x_i,y_j)+\varphi(x_{i+1},y_j)}{(\Delta x)^2}-\frac{\varphi(x_i,y_{j-1})-2\varphi(x_i,y_j)+\varphi(x_i,y_{j+1})}{(\Delta y)^2}=s(x_i,y_i) \tag{5}$$

Several equations from (5) would be:

| i | j | equation |
|---|---|---|
| 1 | 1 | $-\frac{\varphi_{0,1}-2\varphi_{1,1}+\varphi_{2,1}}{(\Delta x)^2}-\frac{\varphi_{1,0}-2\varphi_{1,1}+\varphi_{1,2}}{(\Delta y)^2}=s(x_1,y_1)$ |
| 2 | 1 | $-\frac{\varphi_{1,1}-2\varphi_{2,1}+\varphi_{3,1}}{(\Delta x)^2}-\frac{\varphi_{2,0}-2\varphi_{2,1}+\varphi_{2,2}}{(\Delta y)^2}=s(x_2,y_1)$ |
| ⋮ | ⋮ | ⋮ |
| N-1 | 1 | $-\frac{\varphi_{N-2,1}-2\varphi_{N-1,1}+\varphi_{N,1}}{(\Delta x)^2}-\frac{\varphi_{N-1,0}-2\varphi_{N-1,1}+\varphi_{N-1,2}}{(\Delta y)^2}=s(x_{N-1},y_1)$ |
| ⋮ | ⋮ | ⋮ |
| 1 | 2 | $-\frac{\varphi_{0,2}-2\varphi_{1,2}+\varphi_{2,2}}{(\Delta x)^2}-\frac{\varphi_{1,1}-2\varphi_{1,2}+\varphi_{1,3}}{(\Delta y)^2}=s(x_1,y_2)$ |
| 1 | N-1 | $-\frac{\varphi_{0,N-1}-2\varphi_{1,N-1}+\varphi_{2,N-1}}{(\Delta x)^2}-\frac{\varphi_{1,N-2}-2\varphi_{1,N-1}+\varphi_{1,N}}{(\Delta y)^2}=s(x_1,y_{N-1})$ |
| ⋮ | ⋮ | ⋮ |

Table 1: Finite difference equations for 2D Poisson equation

It is easily seen from Table 1 that we will end up with a large system of linear equations, $N_xN_y$ equations and unknowns since there is an equation for every combination of $x$ and $y$. If you look closely, each equation only has non-zero coefficients for at most 5 terms (some lost to boundary conditions). They happen to also follow a pattern with a point at the center of a 5-point star as shown in Figure 2. Thus, the coefficient matrix is a banded diagonal matrix with 5 non-zero elements on most rows.

If we relabel the center point $(x_i,y_i)$ and assign a unique integer label $n=(i-1)N_y+j$, the

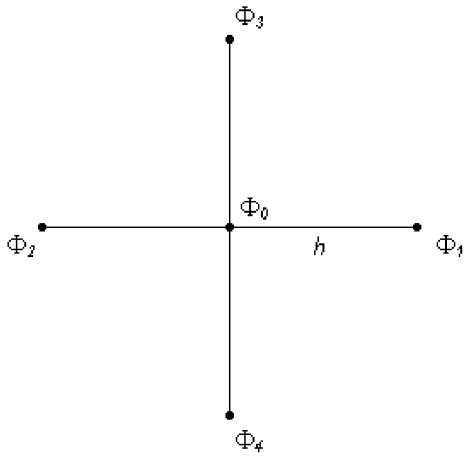



Figure 2: Geometry of the five-point star used in 2D finite differences

neighboring points are:

| | | |
|---|---|---|
| north (N) | $(x_i, y_{j+1})$ | $m = n + 1$ |
| east (E) | $(x_{i+1}, y_j)$ | $m = n + N_y$ |
| south (S) | $(x_i, y_{j-1})$ | $m = n - 1$ |
| west (W) | $(x_{i-1}, y_j)$ | $m = n - N_y$ |

We will compute the answer vector of $\varphi$ at the grid points:

$$\boldsymbol{\varphi} = \left[\varphi_n\right] \in \mathbb{R}^N \qquad N = N_x N_y \qquad \varphi_n = \varphi\left(x_i, y_j\right) \qquad n = (i-1)\, N_y + y \tag{6}$$

For each grid point on the boundary (circled in Figure 1), the boundary condition is enforced via the corresponding linear algebraic equation:

$$\begin{array}{cc} \varphi_n = \varphi\left(x_i, y_i\right) = 0 & n = (i-1)\, N_y + j \\ \text{BC1:} \quad i = 1 & 1 \le j \le N_y \\ \text{BC2:} \quad i = N_x & 1 \le j \le N_y \\ \text{BC3:} \quad 1 \le i \le N_x & j = 1 \\ \text{BC4:} \quad 1 \le i \le N_x & j = N_y \end{array}$$

The resulting linear system when using this optimum 5-point start labeling scheme is:

$$A_{n,n-N_y}\varphi_{n-N_y} + A_{n,n-1}\varphi_{n-1} + A_{n,n}\varphi_n A_{n,n+1}\varphi_{n+1} + A_{n,n+N_y}\varphi_{n+N_y} = b_n \tag{7}$$

$$A_{n,n-N_y} = A_{n,n+N_y} = \left[\frac{-1}{(\Delta x)^2}\right], \quad A_{n,n-1} = A_{n,n+1} = \left[\frac{-1}{(\Delta y)^2}\right] \tag{8}$$

$$A_{nn} = \left[\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2}\right], \quad b_n = s\left(x_i, y_j\right) \tag{9}$$

An example with numbers (do not worry about their values) is:

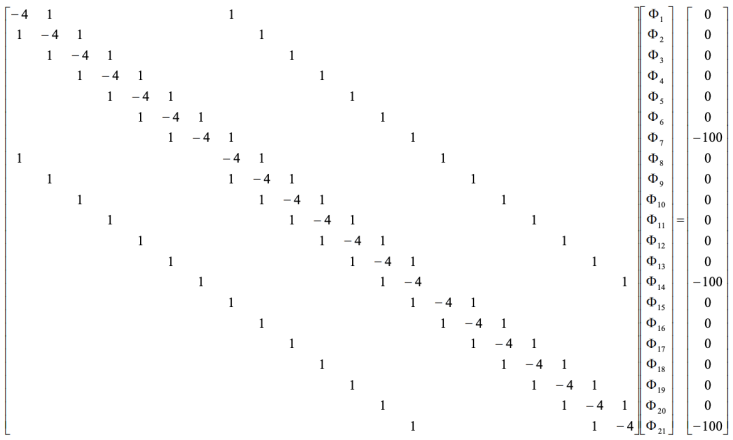


Figure 3: Example linear system for 2D Poisson equation

**Multidimensional boundary value problems deliverables**

Write a function called `poisson_2d.m` which accepts a function name that gives the values of $s(x_i, y_j)$, the length and height of the domain, and the number of points in both the $x$ and $y$ directions and uses finite differences to determine $\varphi(x_i, y_j)$ for all $x$ and $y$ in the domain. The usage of the function will be:

```
[PHI, S, X, Y, soln_time, A_mem] = poisson_2d(src_fun_name, L, H, num_pts);
```

Write a script called `test_poisson_2D.m` which runs the finite difference computations and plots the source function $s(x_i, y_j)$ as well as $\varphi(x_i, y_j)$ for each set of source functions below:

1. Zero source - $s(x_i, y_j) = 0$
2. Uniform source - $s(x_i, y_j) = 1$
3. Linear source in $x$ varying from 0 to 1 - $s(x_i, y_j) = \frac{x}{L}$
4. Linear source in both $x$ and $y$ - $s(x_i, y_j) = \frac{x}{L} + \frac{2y}{H}$
5. Non-linear source in both $x$ and $y$ - $s(x_i, y_j) = \left(\frac{x}{L}\right)^2 + \sin(10\frac{y}{H})$

A template for `poisson_2D.m` and complete uniform source `source_uniform_2D.m` function will be provided. The report must include a discussion on the different source functions and how the effect the final solution. Does the solution make sense given each source function? Please explain.

Example output from `Possion_2D.m` with a uniform source at two different number of grid points is shown Figure 5. Notice how the solution changes slightly when the number of grid points increased. The solution to the PDE in (2) **cannot** be dependent on the grid size. Thus, it is required of the user to find the finest grid size that provides an accurate solution but not too fine that the computation takes an enormous amount of time.

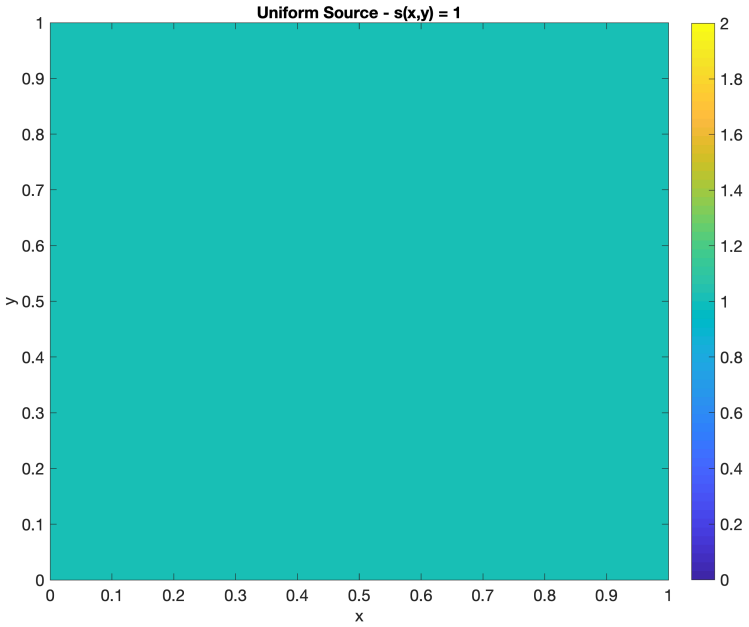
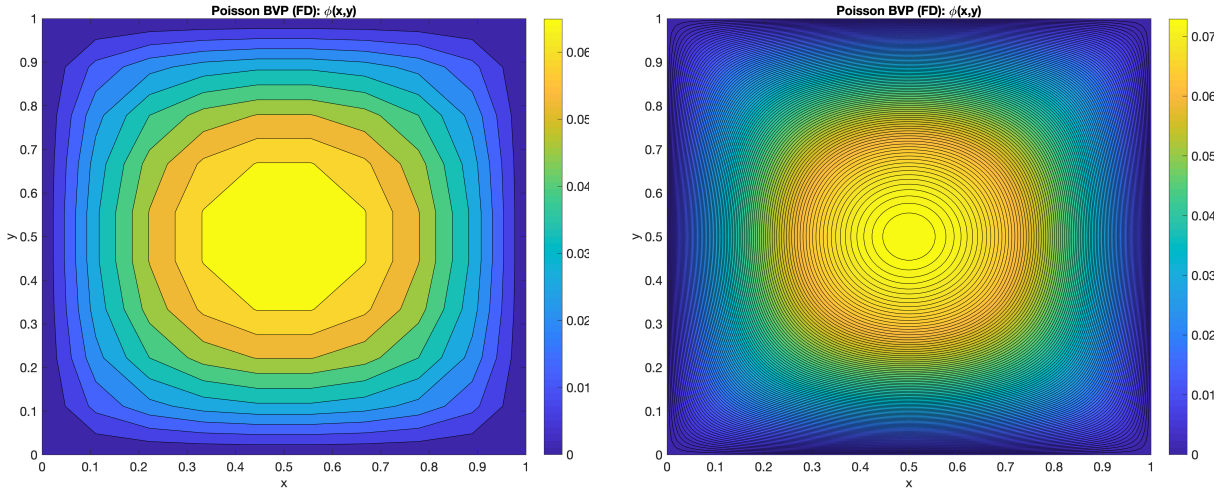



Figure 4: Example uniform source with $s(x, y) = 1$




(a) Low resolution ($N_x = N_y = 10$)

(b) High resolution ($N_x = N_y = 100$)

Figure 5: Example $\varphi(x, y)$ with uniform source at several grid resolutions

The method to find this optimum is called *mesh refinement* and is described below:

1. Start with an initial grid size (i.e. $N_x = 10$, $N_y = 10$) then solve using finite differences
2. Double the number of grid points (i.e. $N_x = 20$, $N_y = 20$) then solve using finite differences
   (a) If the solution changes more than a preset tolerance (say a relative tolerance of $1 \times 10^{-3}$) then double the number of grid points again
   (b) If the solution is within the preset tolerance, you are done or can decrease the number of grid points if you intend to simulate again

For **Extra Credit**, implement this mesh refinement method in a new function called `poisson_2D_mesh_refine.m` which is called the previous `poisson_2D.m` but uses the `num_pts` input as the initial mesh count and carries out mesh refinement during the solution. The resulting `num_pts` as well as the computed relative tolerance should be outputs of this function in addition to the others for `poisson_2D.m`. Also create a test script `test_poisson_2D_mesh_refine.m` which tests your mesh refinement function using a single source function. Be sure to demonstrate it's use (i.e. capture the mesh refinement process showing the progression of the error and solution along the way). You will need to come up with a way to compare the two solutions.

### Beyond homogeneous Dirichlet boundary conditions

All discussion above has been with all homogeneous Dirichlet boundary conditions with $\varphi = 0$ at all boundaries. However, the physical world requires a wider variety of boundary conditions in order to mathematically model the physical systems we encounter. A listing of the most common boundary conditions include:

| **Name** | **Equation** | **What it is...** |
|---|---|---|
| Homogeneous Dirichlet | $\varphi(0,y) = 0$ | Dependent variable is value |
| Non-homogeneous Dirichlet | $\varphi(0,y) = f(0,y)$ | Dependent variable is given by a function |
| Homogeneous von Neumann | $\left.\frac{\partial\varphi}{\partial x}\right\|_{0,y} = 0$ | Flux is zero |
| Non-homogeneous von Neumann | $\left.\frac{\partial\varphi}{\partial x}\right\|_{0,y} = f(0,y)$ | Flux is given by a function of space only |
| Homogeneous Robin | $\left.\frac{\partial\varphi}{\partial x}\right\|_{0,y} + A\varphi(0,y) = 0$ | Flux is given by a function of space and $\varphi$ whose linear combination equates to zero |
| Non-homogeneous Robin | $\left.\frac{\partial\varphi}{\partial x}\right\|_{0,y} + A\varphi(0,y) = f(0,y)$ | Flux is given by a function of space and $\varphi$ whose linear combination equates to a function of space |

Table 2: Types of boundary conditions with equations shown to be evaluated at the boundary with $x = 0$ and all $y$

Thus, we must figure out how to handle other types of boundary conditions. Non-homogeneous Dirichlet boundary conditions are easy to implement by changing (9) to not only include $s(x_i, y_i)$ but also the value of $\varphi(x_i, y_j)$ at that point but **only** at the boundaries. Thus equation (9) becomes:

$$b_n\Big|_{\text{boundary}} = \varphi(x_i, y_j) = g(x_i, y_j) \tag{10}$$

### Beyond homogeneous Dirichlet boundary conditions deliverables

Enhance the `poisson_2d.m` function to include the ability of handling all types of Dirichlet boundary conditions. Call this function `poisson_2d_dirichlet.m` which is essentially the same as the

`poisson_2d.m` function but with one additional input in which a single function name is passed called `dirichlet_bcs.m` which takes in $x$, $y$, $L$, and $H$ and returns the value of the boundary condition ($\varphi$) at that particular $x$ and $y$. Write a test script `test_poisson_2D_Dirichlet.m` which runs a single finite different computation with the uniform source described above and the following boundary conditions:

$$\varphi(0, y) = 1 \qquad \varphi(L, y) = y \qquad \varphi(x, 0) = -2 \qquad \varphi(x, H) = x^2 \tag{11}$$

Be sure to plot the source, $\varphi(x, y)$ and the Dirichlet boundary conditions verifying they have taken on the values in (11).