DD2370 Computational Methods for Electromagnetics Application: *Finite Difference*

Stefano Markidis, KTH Royal Institute of Technology

Maxwell Equations in the Electrostatic Limit

- **Current and change of current** are small enough **not to** generate a significative magnetic field and inductive electric field
- In this case, we use only **Gauss law** instead of the full set of equations
- And use the definition E as – gradient of potential

$$\nabla \cdot E = \rho / \epsilon$$ Gauss law

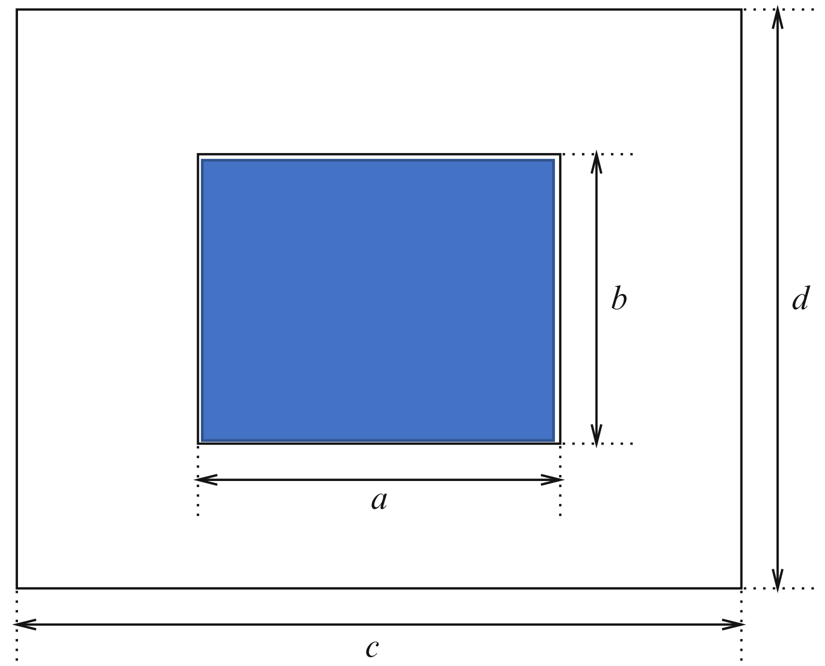
$$\downarrow$$

$$\nabla \cdot (-\nabla \Phi) = -\nabla^2 \Phi = \rho / \epsilon$$

Electrostatic potential is proxy for E

Poisson Equation in Coaxial Transmission Line
- and then Capacitance

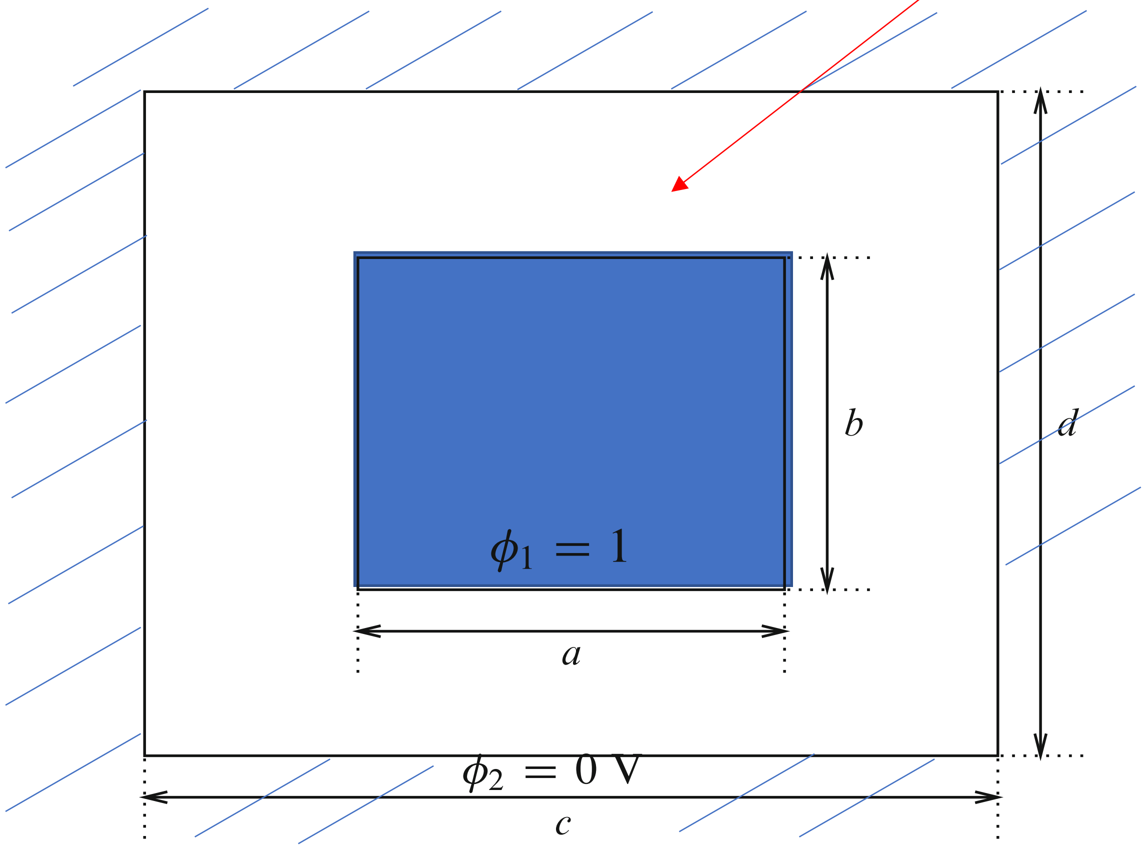
1. Assign Potential in the inner and outer boundary
2. Calculate the Electrostatic Potential from solving **Poisson Equation by finite difference**
    1. Calculate the charge per unit length from Potential
    2. Calculate the Capacitance per unit length from charge per unit length



Solve Poisson Equation

$$\nabla^2\phi = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} = 0$$

Between the inner and outer conductors



2D Discretization of Poisson Equation

- This equation applies for all internal points *(x_i, y_j)* on the grid.
- As **boundary conditions**, we *Phi = 0* on the outer conductor and *Phi = 1* on the inner conductor
- Compute the charge per unit length *Q* from the solution.
- Then the capacitance per unit length is *C = 1/Q*, since the voltage across the capacitor is 1

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \approx \frac{f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1} - 4 f_{i,j}}{h^2} = 0$$

Jacobi and Gauss-Seidel Iteration

- Jacobi and Gauss–Seidel iteration, to solve the discretized Poisson Equation
  - It doesn't require that the system of linear equations be formed and stored explicitly
- **They are iterative method**. They starts with an initial guess for the solution $f_{i,j}$ at all internal grid points, e.g., $\boldsymbol{f_{i,j} = 0}$
- The iterative method then updates these values until we reach a converged solution that satisfies the finite difference approximation at all internal grid points.
- ***BC:*** $f_{i,j}$ is set to its prescribed values on the boundaries, where the solution is known from the boundary conditions, and these values are kept fixed.

Jacobi Iteration

The Jacobi iteration for the Poisson Equation is

$$f_{i,j}^{(n+1)} = \frac{1}{4}\left(f_{i-1,j}^{(n)} + f_{i+1,j}^{(n)} + f_{i,j-1}^{(n)} + f_{i,j+1}^{(n)}\right)$$

n = iteration

At every grid point, the potential is the **average of the potential** at the four nearest neighbors.

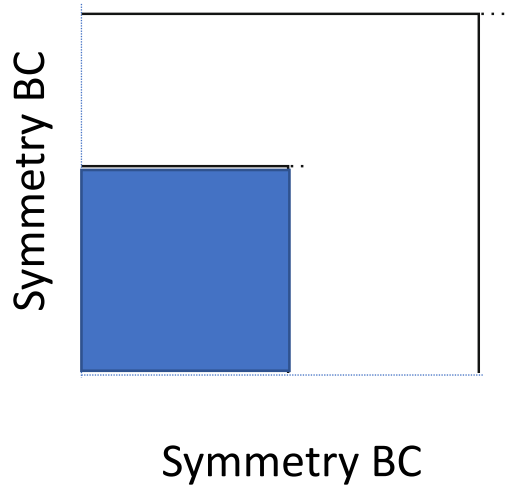
Jacobi → Gauss-Seidel Iteration

- Jacobi gives very **slow convergence**
- One modification is the so-called **Gauss–Seidel** iteration, where the "old" values of *f* are immediately overwritten by new ones, as soon as they are computed.
- If *f* is updated in the order of increasing i and j , the Gauss–Seidel scheme is

$$f_{i,j}^{(n+1)} = \frac{1}{4}\left(f_{i-1,j}^{(n+1)} + f_{i+1,j}^{(n)} + f_{i,j-1}^{(n+1)} + f_{i,j+1}^{(n)}\right)$$

Computing Poisson Equation in ¼ Domain

- Generate a grid such that the conducting boundaries fall on the grid points.
  - Exploit the **symmetry** and compute only on the upper right quarter, to reduce the number of unknowns.
    - Around a line of symmetry with a constant ,we enforce the symmetry by $f_{i+n,j} = f_{i-n,j}$, where *n* is a positive integer.
      - Symmetry lines with a constant j are treated analogously
- Introduce the **boundary conditions** by setting $f_{i,j} = 0$ on the outer conductor and $f_{i,j} = 1$ on the inner conductor.
- Iterate with the Gauss–Seidel scheme over the internal points to solve for the potential.



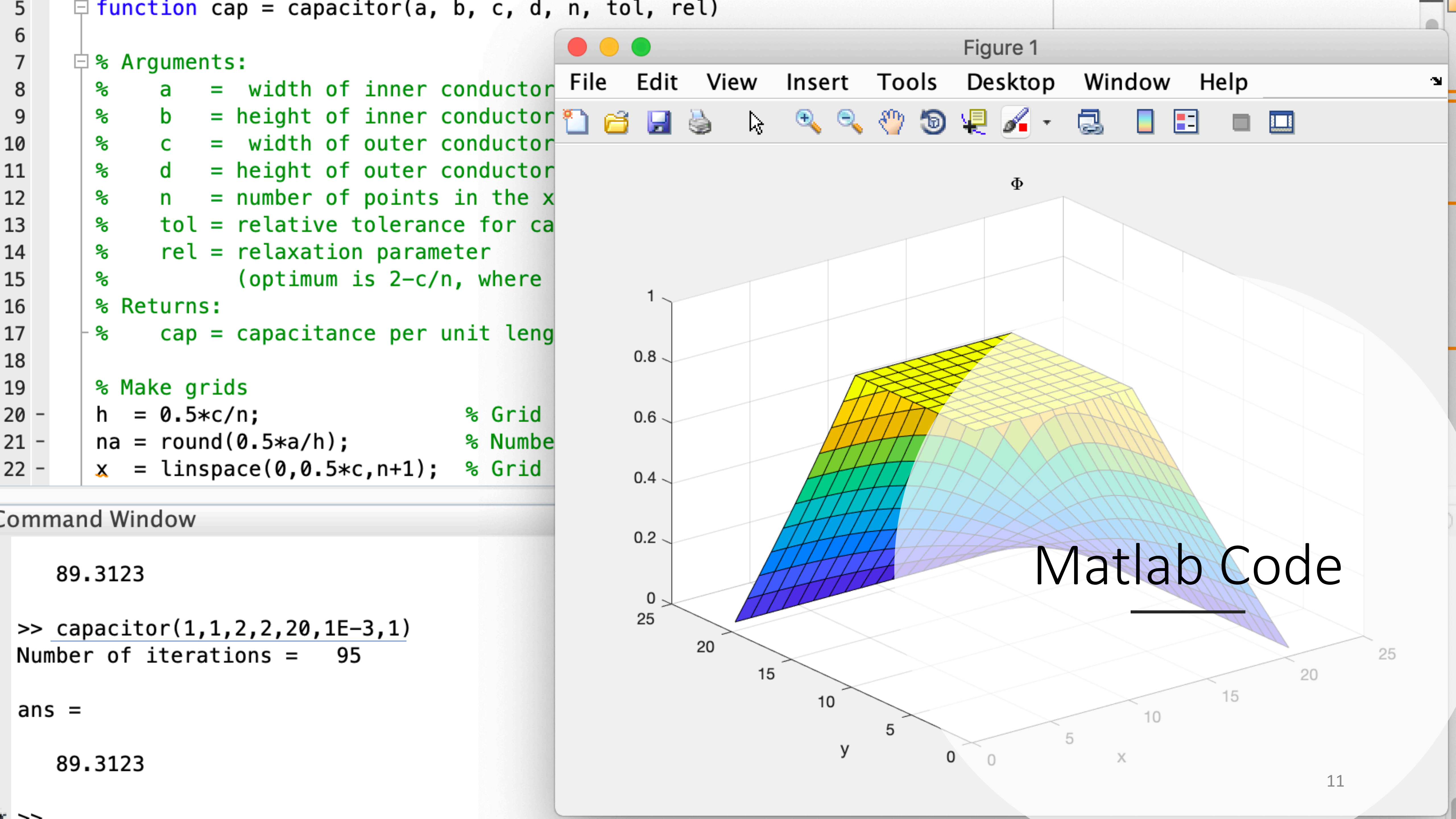
Computing Charge and Capacitance

The capacitance per unit length is C= **Q**/V. The charge on the inner conductor **Q** can be computed from Gauss's law

$$Q = \epsilon_0 \oint \boldsymbol{E} \cdot \hat{\boldsymbol{n}}\, dl = -\epsilon_0 \oint \frac{\partial \phi}{\partial n} dl$$

The capacitance per unit length is C= **Q**/V.

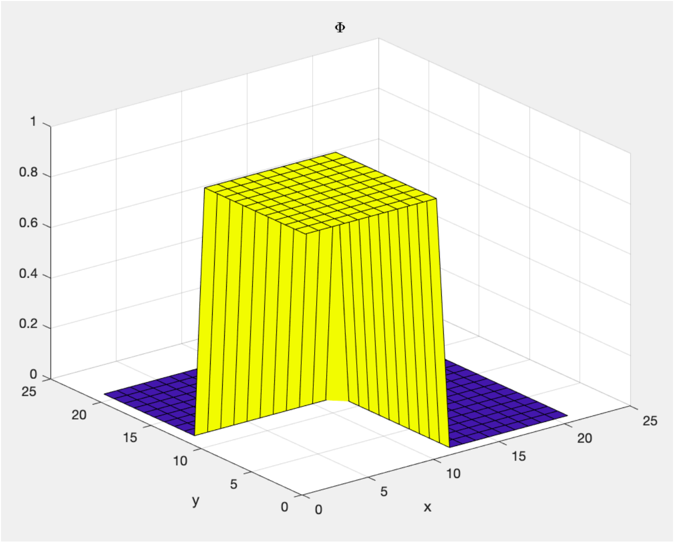
Matlab Code



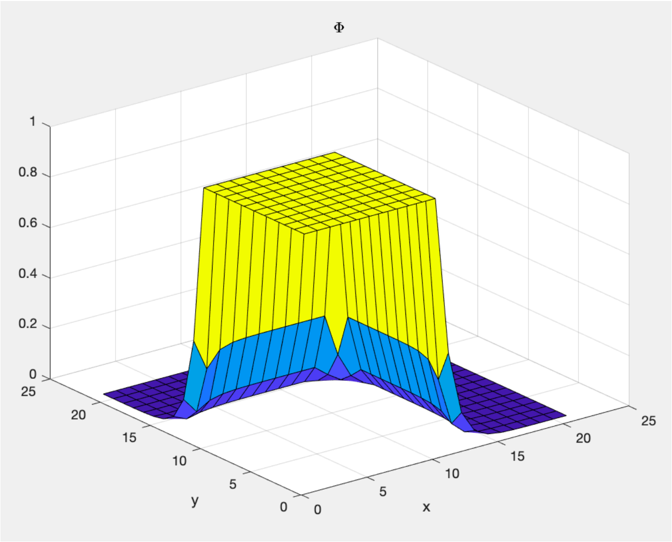
function cap = capacitor(a, b, c, d, n, tol, rel)
% Arguments:
% a = width of inner conductor
% b = height of inner conductor
% c = width of outer conductor
% d = height of outer conductor
% n = number of points in the x
% tol = relative tolerance for ca
% rel = relaxation parameter
% (optimum is 2-c/n, where
% Returns:
% cap = capacitance per unit leng
% Make grids
h = 0.5*c/n; % Grid
na = round(0.5*a/h); % Numbe
x = linspace(0,0.5*c,n+1); % Grid
Command Window
89.3123
>> capacitor(1,1,2,2,20,1E-3,1)
Number of iterations = 95
ans =
89.3123
>>
Figure 1
File Edit View Insert Tools Desktop Window Help
Φ
1
0.8
0.6
0.4
0.2
0
25
20
15
10
5
0
y
0
5
10
15
20
25
x

Matlab Code *capacitor.m* available in Canvas

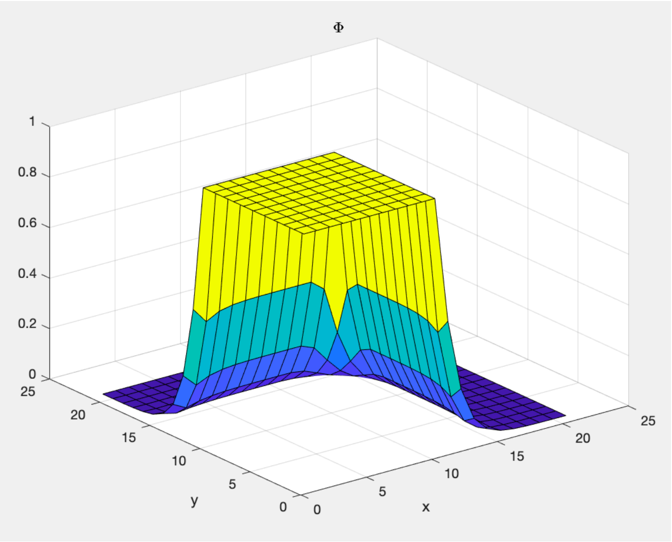
n=1



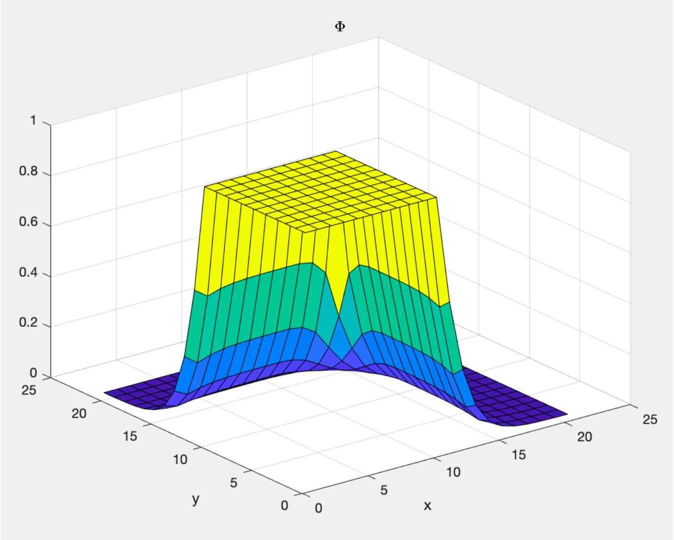
n=2



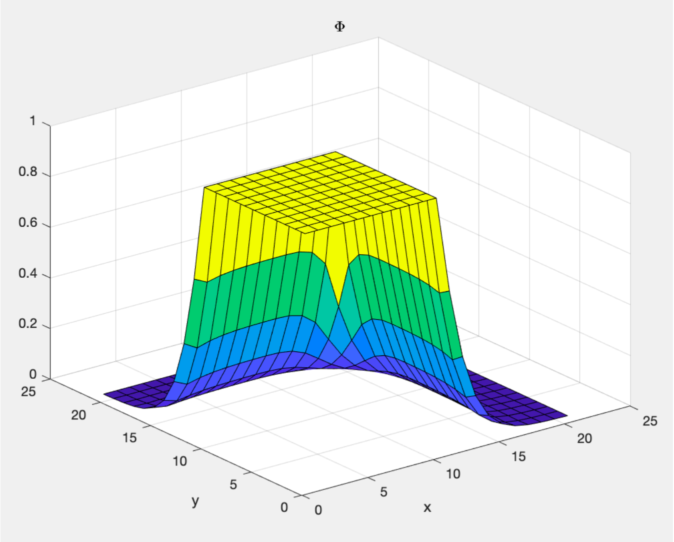
n=3



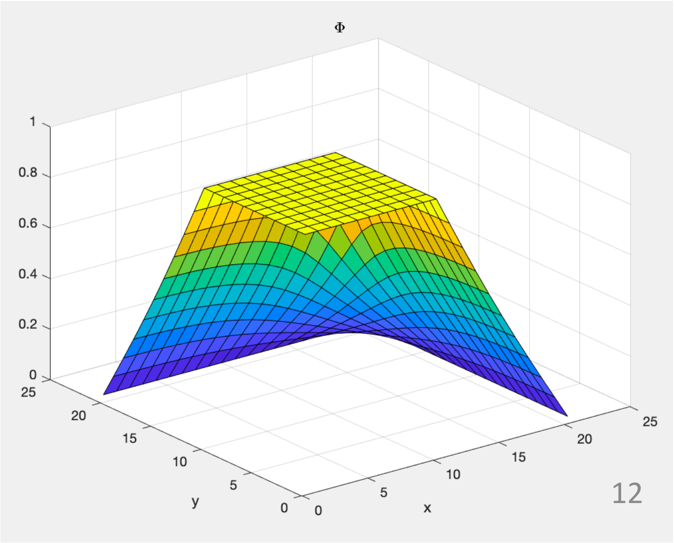
n=4



n=5



n=80



Assignment

- Convergence Test
- Formulate as Linear Solver problem using matrix formulation