Matlab:

Please follow the link given below and everything regarding installation of matlab is provided there. After installation before you run it make sure that you are logged in the network license server from matlab.iitb.ac.in. If you guys face any difficulties then please let me know.

Link: : ftp://ftp.iitb.ac.in/IITB_private/Matlab/MATHWORKS_R2018a/

Finite Difference Scheme:

The principle of finite difference methods is close to the numerical schemes used to solve ordinary differential equations. It consists in approximating the differential operator by replacing the derivatives in the equation using differential quotients. The domain is partitioned in space and in time and approximations of the solution are computed at the space or time points. The error between the numerical solution and the exact solution is determined by the error that is committed by going from a differential operator to a difference operator. This error is called the discretization error or truncation error. The term truncation error reflects the fact that a finite part of a Taylor series is used in the approximation.

For the sake of simplicity, let us take one dimensional case. The 1st derivative is found from Taylor Series expansion as:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(\varepsilon) + \frac{h^3}{3!}f'''(x),$$

$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x)$…………..(i)

and

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x),$$

$f'(x) = \frac{f(x)-f(x-h)}{h} + \frac{h}{2!}f''(x) - \frac{h^2}{3!}f'''(x)$…………..(ii)

Where the second and third term in the equation (i) and (ii) are the error terms.

The approximation of the derivative at x that is based on the values of the function at x-h and x, i.e.,

$f'(x) \approx \frac{f(x)-f(x-h)}{h}$…………………......(iii)

is called the backward difference, based on the values of the function at x+h and x

$f'(x) \approx \frac{f(x+h)-f(x)}{h}$…………………......(iv)

is called the forward difference.

To get a better definition of first derivative one can formulate it based on cantered difference formula from points f(x-h) and f(x+h)

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

Based on equation (i) and (ii) we can write

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + \frac{h^2}{3!} f'''(x) + \cdots$$

Now assuming the truncation error is negligible we come up with

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Similarly we can get the second derivative by subtracting f(x+h) and f(x-h) from equation (i) and (ii)

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

Solving Differential Equation Using Finite Difference Scheme:

a) First Order Differential Equation:
The electric field inside the material can be represented using the first order differential equation given below.

$$\frac{d\xi(x)}{dx} = \frac{\rho(x)}{\epsilon}$$

Using the finite difference scheme we can convert the above differential equation into a set of linear equations:

$$\frac{\xi(x_{i+1}) - \xi(x_i)}{\Delta x} = \frac{\rho(x_i)}{\epsilon}$$

$$\Rightarrow \xi(x_{i+1}) - \xi(x_i) = \frac{\Delta x\, \rho(x_i)}{\epsilon}$$

$$\Rightarrow \xi_{i+1} - \xi_i = \frac{\Delta x\, \rho_i}{\epsilon}$$

So the set of linear equations can be constructed by varying 'i' which is nothing but the mesh position. We can represent the set of linear equations using matrix.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{[M]} \times \underbrace{\begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \\ \xi_5 \\ \vdots \\ \xi_{n-1} \\ \xi_n \end{bmatrix}}_{[X]} = \frac{\Delta x}{\epsilon} \underbrace{\begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_5 \\ \vdots \\ \rho_{n-1} \\ \rho_n \end{bmatrix}}_{[N]}$$

To find out the solution for the above mentioned equation we need to find out the inverse of Matrix [M] and multiply it with [N] matrix. That is going to give us the solution in a vector format which is [X]. i.e.

$$[X] = [M]^{-1}[N]$$

b) Second Order Differential Equation:

The most important equation for the study of any electrostatic problem is Poisson's equation. Poisson's equation is a second order differential equation shown below.

$$\frac{d^2\emptyset(x)}{dx^2} = -\frac{\rho(x)}{\epsilon}$$

Using the finite difference scheme we can convert the above differential equation into a set of linear equations:

$$\frac{\emptyset(x_{i-1}) - 2\emptyset(x_i) + \emptyset(x_{i+1})}{(\Delta x)^2} = \frac{\rho(x_i)}{\epsilon}$$

$$\emptyset_{i-1} - 2\emptyset_i + \emptyset_{i+1} = (\Delta x)^2 \frac{\rho_i}{\epsilon}$$

So the set of linear equations can be constructed by varying 'i' which is nothing but the mesh position. We can represent the set of linear equations using matrix.

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}_{[M]} \times \underbrace{\begin{bmatrix} \emptyset_1 \\ \emptyset_2 \\ \emptyset_3 \\ \emptyset_4 \\ \emptyset_5 \\ \emptyset_6 \\ \vdots \\ \emptyset_{n-1} \\ \emptyset_n \end{bmatrix}}_{[X]} = \frac{(\Delta x)^2}{\epsilon} \underbrace{\begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \rho_4 \\ \rho_5 \\ \rho_6 \\ \vdots \\ \rho_{n-1} \\ \rho_n \end{bmatrix}}_{[N]}$$

To find out the solution for the above mentioned equation we need to find out the inverse of Matrix [M] and multiply it with [N] matrix. That is going to give us the solution in a vector format which is [X]. i.e.

$$[X] = [M]^{-1}[N]$$

I am providing here an example for solving Poisson's equation. Assume two metal plates A and B are kept at a separation of 100um in free space. Plate A is grounded (at x=0) and while plate B is held at a potential of 1V (at x=100um). Find the potential profile from Plate A to Plate B.

Solution: So according to the finite difference scheme described above we can discretise the space between plate A and plate B and come up with a set of linear equations. The matrix will look like:

$$[M] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$[X] = \begin{bmatrix} \emptyset_1 \\ \emptyset_2 \\ \emptyset_3 \\ \emptyset_4 \\ \emptyset_5 \\ \emptyset_6 \\ \vdots \\ \emptyset_{n-1} \\ \emptyset_n \end{bmatrix}$$

Now there is no space charge in between the two plate, so all the elements of matrix [N] are zero except two boundary points the first element and the last element of the matrix. We need to provide the boundary values. Assume plate A is at x=0 position and it is mentioned that the plate is grounded so its potential will be '0'. So the first element of [N] matrix is '0' and the last element is '1' as plate B is connected to '1' volt potential.

$$[N] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

So the final set of matrices:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \emptyset_1 \\ \emptyset_2 \\ \emptyset_3 \\ \emptyset_4 \\ \emptyset_5 \\ \emptyset_6 \\ \vdots \\ \emptyset_{n-1} \\ \emptyset_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Now we can find out the values for potential at each position by determining the elements of vector $[X] = [M]^{-1}[N]$.

I am providing the matlab code for an example to help you at the beginning.

```
clear;
clc;
nx=201;                    % Number of grid points %
M=sparse(nx,nx);           % Creates nx*nx zero sparse matrix %
d=100e-4;                  % Distance between plates in cm %

% Finite difference method to solve 2nd order derivative in Poisson's
equation %
for i=2:nx-1
M(i,i-1)=1; M(i,i+1)=1; M(i,i)=-2;
end
M(1,1)  = 1;               % Setting boundary conditions %
M(nx,nx)  = 1;
N=zeros(nx,1);             % Setting N matrix of MX=N linear equation %
N(nx)=1;
X=M\N;                     % Finding solutions %
NX=linspace(0,d,nx);
plot(NX,X,'linewidth',2);
grid on;
temp2=['Potential profile from plate A to plate B of Problem No. 4.(a)'];
title(temp2,'fontsize',14);
xlabel('Distance (in cm)','fontsize',14);
ylabel('Potential (in Volt)','fontsize',14);
```

Newton’s method to solve a system of non-linear equations:

Usually one can find the solution of a system of equation if the number of unknown matches with the number of equations. So the system of “n” variables can be expressed as:

$$\begin{gathered} f_1(x_1, x_2, x_3, \dots\dots\dots\dots.., x_n) = 0 \\ f_2(x_1, x_2, x_3, \dots\dots\dots\dots.., x_n) = 0 \\ f_3(x_1, x_2, x_3, \dots\dots\dots\dots.., x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, x_3, \dots\dots\dots\dots.., x_n) = 0 \end{gathered}$$

For single valued case the Newton’s method can be derived by considering linear approximation of the function “f” at the initial guess x_0. From Calculus, the following is the linear approximation of function “f” at x_0, for vector and vector-valued functions

$$f(x) = f(x_0) + Df(x_0)(x - x_0) + \cdots$$

Here, $Df(x_0)$ is a N*N matrix which is formed by taking partial derivative of components of $f(x_0)$

$$Df(x_0) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_0) & \frac{\partial f_1}{\partial x_2}(x_0) & \frac{\partial f_1}{\partial x_3}(x_0) & \dots\dots\dots\dots\dots & \frac{\partial f_1}{\partial x_n}(x_0) \\ \frac{\partial f_2}{\partial x_1}(x_0) & \frac{\partial f_2}{\partial x_2}(x_0) & \frac{\partial f_2}{\partial x_3}(x_0) & \dots\dots\dots\dots\dots & \frac{\partial f_2}{\partial x_n}(x_0) \\ \frac{\partial f_3}{\partial x_1}(x_0) & \frac{\partial f_3}{\partial x_2}(x_0) & \frac{\partial f_3}{\partial x_3}(x_0) & \dots\dots\dots\dots\dots & \frac{\partial f_3}{\partial x_n}(x_0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1}(x_0) & \frac{\partial f_n}{\partial x_2}(x_0) & \frac{\partial f_n}{\partial x_3}(x_0) & \dots\dots\dots\dots\dots & \frac{\partial f_n}{\partial x_n}(x_0) \end{pmatrix}$$

To find out the value of x which makes the value of f(x) equal to zero vectors. Let choose x_1 so that

$$\begin{gathered} f(x_0) + Df(x_0)(x_1 - x_0) = 0 \\ x_1 = x_0 - (Df(x_0))^{-1} f(x_0) \end{gathered}$$

Provided that the inverse matrix does exists. To find out the solution this above method can be run iteratively and at each iteration the value of x_1 will be updated to the desired tolerance error limit.

Example:

Suppose that the two given equations are: $x_1 + 2x_2 = 2$; $x_1^2 + 4x_2^2 = 8$ and we have to find out the solution for x_1 and x_2.

```
clear;
clc;

% Graphical Solution %
```

```
x1=-2:0.01:10;
x2=(2-x1)/2;        % First funtion %
x3=sqrt(8-x1.^2)/2; % Second function %
figure(1)
plot(x1,x2,'b',x1,x3,'g','linewidth',2);
grid on;
temp1=['Finding an approximate solution of x1 and x2 through graphical
method'];
title(temp1,'fontsize',14);
xlabel('x1','fontsize',14);
ylabel('f_1(x_1, x_2) and f_2(x_1, x_2)','fontsize',14);
legend('f_1(x_1, x_2) = x_1 + x_2 - 2','f_2(x_1, x_2) = x_1^2 + 4x_2^2 -
8','fontsize',14);

% Finding Solutions Using Newton's method %
format long
f=inline('[x(1)+(2*x(2))-2 ; (x(1)^2)+(4*x(2)^2)-8]');
t1 = input('Insert the approximate value of x1 from the figure...');
t2 = input('Insert the approximate value of x2 from the figure...');
x=[t1;t2];
iter=10;
Df=inline('[1 , 2 ; 2*x(1) , 8*x(2)]');
for i = 1:iter
x = x + -Df(x)\f(x);
error1(i)=max(abs(f(x)));
end
fprintf('The solutions from Nonlinear Newtons method for x1....%f',x(1));
fprintf(' and for x2....%f',x(2));
figure(2)
plot(error1,'b','LineWidth',2)
temp2=['Rate of convergence to find the solution using Nonlinear Newtons
method (x_1 = -0.732051 and x_2 = 1.366025)'];
title(temp2,'fontsize',14);
xlabel('Number of Iteration','fontsize',14);
ylabel('Absolute Error','fontsize',14);
```