```
close all;
clear all;
% This codes solves the 2D Poisson/Laplace equation: u_xx+u_yy=C
% for Dirichlet and Neumann boundary conditions
% on a rectangular dmain
% Domain size
Lx = 3;
Ly = 2;

% Boundary conditions
% Boundary type (D for Dirichlet, N for Neumann):
% type: (no-slip condition)
Tleft_type = 'D';
Tright_type = 'D';
Ttop_type = 'D';
Tbottom_type = 'D';
% value: (Poisson - no-slip condition and Dirichlet) (Laplace - Need at least 1 th
% non zero)
Tleft = @(x,y) 0;
Tright = @(x,y) 0;
Ttop = @(x,y) 0;
Tbottom = @(x,y) 0 ;

% Source term
dpdz = -1; % Pressure gradient is negative (change to 0 for Laplace)

% Number of grid points in x and y directions
Nx = 61;
Ny = 41;

% Create the mesh
x = linspace(0,Lx,Nx);
y = linspace(0,Ly,Ny);

% Step size in x and y directions
dx = Lx / ( Nx - 1 );
dy = Ly / ( Ny - 1 );

%Adding the fictitious points:
x=[x(1)-dx/2 x x(end)+dx/2];
y=[y(1)-dy/2 y y(end)+dy/2];

% The field on the grid
T = zeros( Nx+2, Ny+2 );

% Set the boundary conditions
% left, x = 0 boundary
if Tleft_type == 'D';
    T( 2, : ) = Tleft(x,y);
elseif Tleft_type == 'N'
    T(1,:) = T(3,:)-2*dx*Tleft(x,y);
end
```

```
% right, x = Lx boundary
if Tright_type == 'D';
    T( Nx+1, : ) = Tright(x,y);
elseif Tright_type == 'N'
    T(Nx+2,:) = T(Nx,:)-2*dx*Tright(x,y);
end
% bottom, y = 0 boundary
if Tbottom_type == 'D';
    T( :, 2 ) = Tbottom(x,y);
elseif Tbottom_type == 'N'
    T(:,1) = T(:,3)-2*dx*Tbottom(x,y)';
end
% top, y = Ly boundary
if Ttop_type == 'D';
    T( :, Ny+1 ) = Ttop(x,y);
elseif Ttop_type == 'N'
    T(:,Ny+2) = T(:,Ny)-2*dx*Ttop(x,y)';
end
% return;
% Set an initial high value for the convergence error
E = 100;
% Initialize the iteration matrices, T^(k+1)=T2 and T^k=T1
T1 = T;
T2 = T1;
% k = the number of Jacobi iteration
k = 1;
% Continue the Jacobi iteration until the solution has converged
while E>1e-3
    k
    for i = 2:Nx+1
        for j = 2:Ny+1
          % The discrete Laplace equation on a finite difference Stencil
          T2( i, j ) = ((T1(i+1,j)+T1(i-1,j))*dy^2+ ...
              (T1(i,j+1)+T1(i,j-1))*dx^2 - (dpdz*dy^2*dx^2))/(2*dx^2+2*dy^2);
        end
    end
    % updting the B.C.'s:
    % left, x = 0 boundary
    if Tleft_type == 'D';
        T2( 2, : ) = Tleft(x,y);
    elseif Tleft_type == 'N'
        T2(1,:) = T2(3,:)-2*dx*Tleft(x,y);
    end
    % right, x = Lx boundary
    if Tright_type == 'D';
        T2( Nx+1, : ) = Tright(x,y);
    elseif Tright_type == 'N'
        T2(Nx+2,:) = T2(Nx,:)-2*dx*Tright(x,y);
    end
    % bottom, y = 0 boundary
    if Tbottom_type == 'D';
        T2( :, 2 ) = Tbottom(x,y);
    elseif Tbottom_type == 'N'
        T2(:,1) = T2(:,3)-2*dx*Tbottom(x,y)';
    end
    % top, y = Ly boundary
    if Ttop_type == 'D';
        T2( :, Ny+1 ) = Ttop(x,y);
    elseif Ttop_type == 'N'
        T2(:,Ny+2) = T2(:,Ny)-2*dx*Ttop(x,y)';
    end

    % Compute the norm of the error
    E = sqrt(sum(sum(( T2 - T1 ).^2)));
    % Replcae T1 by T2  and increase k by 1 for the next iteration
    T1 = T2;
    k = k+1;
end
```

Plot the solution

```
figure(1);
% Surface plot
[X,Y] = meshgrid(x,y);
imagesc(x,y,T2);
set(gca,'ydir','normal');
colorbar;
xlabel('x','FontSize',20);
ylabel('y','FontSize',20);
clabel = colorbar;
clab = ylabel(clabel,'u(x,y)','fontsize',20,'rot',-90);
set(clab,'units','normalized','position',[4.3 0.5 0]);
set(gca,'FontSize',15);
print -dpdf 'Poisson.pdf';
```

Published with MATLAB® R2014a