```
% 2D Poisson equation with Dirichlet and Neumann boundary conditions
% - Hongli Zhao, UC Berkeley Math 228B

grid_sizes = [10, 20, 40]; % solve problem with 3 different grids

% initialize storage for error from solution
% and for error from numerical integration
err_solution = zeros(1,length(grid_sizes));
err_q = zeros(1,length(grid_sizes));

% parameters
L = 3.0; H = 1.0;

% solve problem with different B values
BB = [0,0.5,1.0];

% we call our function here, with different parameters and grid sizes

% Calculate errors of solution u and integral Q in reference domain

for i = 1:length(BB)
    B = BB(i);
    A = sqrt(0.25 * (L-B)^2 - H^2);

    for k = 1:length(grid_sizes)
        size = grid_sizes(k);
        %[u_err, Q_err] = errPoisson(n,L,B,H);
        % get errors and store
        [error_Q,error_u] = testPoisson(L,B,H,size);

        err_solution(k) = error_u;
        err_q(k) = error_Q;

        % plot particular solution
        if size == 40 && B == 0.5
            % get the actual solutions so that we can plot
            [u, Q, xi, eta] = assemblePoisson(L,B,H,size);

            % map back to physical domain
            x = (A*eta + B/2).*xi;
            y = H*eta;
            % plot solution u

            figure(2);
            subplot(2,2,1)
            surf(x, y, u);
            shading interp
            axis("equal"); colorbar();
            title("u_{physical} 3D");
            shg

            subplot(2,2,2);
```

```
            surf(xi, eta, u);
            shading interp
            axis("equal"); colorbar();
            title("u_{computational} 3D");

            % 2d plots
            subplot(2,2,3)
            contourf(x, y, u, 10);
            axis("equal"); colorbar();
            title("Contour: u_physical");

            subplot(2,2,4)
            contourf(xi, eta, u, 10);
            axis("equal"); colorbar();
            title("Contour: u_computed");
        end
    end

    figure(1)
    subplot(1,3,i)
    loglog(1./grid_sizes, 1./grid_sizes.^2, '--k', ...
        1./grid_sizes, err_solution,'-.r',...
        1./grid_sizes,err_q,'--o','linewidth',2)
    title(['B = ', num2str(B)])
    %set(gca, 'fontsize', 20, 'linewidth',1.75)
    xlabel('h')
    if i==1
        legend('h^2','Error u','Error Q','location', 'se')
    end
    shg
end



function [error_Q, error_u] = testPoisson(L,B,H,n)
    % testPoisson(n) does a few things:
    % computes and returns error arising from 1. computing the PDE
    % ... 2. computing the Q integral
    % solve Poisson with this particular n first
    [u, Q, xi, eta] = assemblePoisson(L,B,H,n);
    % solve exact solution to this Poisson
    [u_exact, Q_exact, xi_exact, eta_exact] = assemblePoisson(L,...
        B,H,80);

    % Error analysis
    % compute error for numerical solution
    error_u =
 compute_inf_norm_error(xi,eta,u,u_exact,xi_exact,eta_exact);

    % compute error for numerical integration Q
    error_Q = compute_int_error(Q,Q_exact);
end
```

```matlab
function [u, Q, xi, eta] = assemblePoisson(L, B, H, n)
% assemblePoisson aims to provide a numerical solution to all
% required parts: 1. solution to PDE, 2. solution to integral

% given parameters and n


% function for assemble the system matrix and RHS.
% Finite difference 9 point stencil.

% returns
% - u that contains all solutions on the unit square, already reshaped
% - Q numerical integral
% - xi
% - eta

    % set up parameters for the matrix and solution vector
    h = 1.0 / n;
    N = (n+1)^2;
    eta = h * (0:n);
    xi = eta;
    % set up the basic coefficients
    D = 0.5 * (L-B);
    A = sqrt(0.25 * (L-B)^2 - H^2);

    % set up the mapping
    umap = reshape(1:N, n+1, n+1)';
    S = zeros(N, N);
    b = zeros(N, 1);
    % set up the spatial coefficients for the 9 point stencil
    % D1,2,3,4 in writeup, depends on xi and eta at (i,j)
    D1 = @(a,b)( (H^2 + A^2 * a^2) / (H^2 * (A*b + 0.5*B)^2));
    D2 = 1/H^2;
    D3 = @(a,b)(-2*A*a)/(H^2 * (A*b+0.5*B));
    D4 = @(a,b)(2*A^2*a)/(H^2*(A*b+0.5*B)^2);

    % fillin the matrix
    for j = 1:n+1
        for i = 1:n+1
            row = umap(i,j);
            % if on the boundaries, we use conditions
            if j == 1 && i >= 1 && i <= n+1 % on bottom i=1:n+1
                S(row, umap(i,j)) = 1;
                b(row) = 0;


            elseif i == n+1 && j >= 2 && j <= n+1 % on right j=2:n+1
                S(row, umap(i,j)) = 1.0;
                b(row) = 0;


            elseif i == 1 && j <= n+1 && j >= 2 % on left j=2:n+1
                S(row, umap(i,j)) = 1.5;
```

```
                S(row, umap(i+1,j)) = -2;
                S(row, umap(i+2,j)) = 0.5;
                b(row) = 0;


            elseif j == n+1 && (i >= 2) && (i <= n) % on top i = 2:n
                % uses 5 point stencil
                dum1 = (A*xi(i))/2;% dummy1, for simplicity
                dum2 = (A*eta(j)+B/2);% dummy2, for simplicity
                S(row, umap(i+1,j)) = dum1;
                S(row, umap(i-1,j)) = -dum1;
                S(row, umap(i,j)) = -1.5 * dum2;
                S(row, umap(i,j-1)) = 2 * dum2;
                S(row, umap(i,j-2)) = -0.5 * dum2;
                b(row) = 0;

            % else, we have the 9-point stencil
            else
                d1 = D1(xi(i),eta(j));
                d2 = D2;
                d3 = D3(xi(i),eta(j));
                d4 = D4(xi(i),eta(j));
                S(row, umap(i,j)) = -(2/h^2)*(d1 + d2);
                S(row, umap(i+1,j)) = d1/h^2 + d4/(2*h);
                S(row, umap(i-1,j)) = d1/h^2 - d4/(2*h);
                S(row, umap(i,j+1)) = d2/h^2;
                S(row, umap(i,j-1)) = d2/h^2;
                S(row, umap(i+1,j+1)) = d3/(4*h^2);
                S(row, umap(i+1,j-1)) = -d3/(4*h^2);
                S(row, umap(i-1,j+1)) = -d3/(4*h^2);
                S(row, umap(i-1,j-1)) = d3/(4*h^2);
                b(row) = -1.0;


            end
        end
    end
   %Sparse storage in Matlab
   S=sparse(S);

   % put numerical solution into unit square
   % as out computational domain indicated
   u = reshape(S\b, n+1, n+1);
   eta_mesh = meshgrid(eta);
   Q = trapz(eta,trapz(xi,u*H.*(A*eta_mesh + B/2),2));
   % meshgrid of reference domain
   [xi,eta] = meshgrid(xi,eta);
end

% helper methods for error computation

function error_u = compute_inf_norm_error(xi_query, eta_query, ...
                u,u_exact, xi_exact,eta_exact)
% Computes inf norm of the vector difference
```

```
    u_exact = interp2(xi_exact,eta_exact,u_exact, ...
            xi_query,eta_query);
    error_u = max(max(abs(u-u_exact)));
end

function error_Q = compute_int_error(Q,Q_exact)
    % computes error of two numbers, which is the abs value
    error_Q = abs(Q-Q_exact);
end
```

Published with MATLAB® R2018b