Question 1

```
function A1q1()

clc

% initialising beta, gamma and n for part a
beta = 2;
gamma = -10;
n = 4;

fprintf('\n    ===========================================================')
fprintf('\n      Question 1a) beta = %2.4f, gamma = %2.4f and n = %3i\n',beta,gamma,n)
fprintf('    ===========================================================\n')

% Creating discrete Convection Diffution system
A = ConvectionDiffusion(beta,gamma,n);
Achen = lcd(beta,gamma,n);

% Checking the that matrix A and Achen are the same
fprintf('\n\n      Inf-norm of A-Achen = %1.4e\n',norm(A-Achen,inf));

% Checking that matrix-vector product routine is correct for 10 random
% vectors
fprintf('\n      Inf-norm error between MatVec and inbuild matrix-vector\n')
fprintf('                          product routine:\n\n')
for j = 1:10
    x = rand(n^2,1);
    y = MatVec(A,x);
    fprintf('            Test %2.0i: error = %1.4e\n',j,norm(y-A*x));
end


% initialising beta, gamma and n for part b
beta = 0;
gamma = 0;
n = 10;

fprintf('\n    ===========================================================')
fprintf('\n      Question 1b) beta = %2.0f, gamma = %2.0f and n = %3i\n',beta,gamma,n)
fprintf('    ===========================================================\n')

% producing 2D Laplacian matix
A = ConvectionDiffusion(beta,gamma,n);

% Creating random RHS vector
```

```
b = rand(n^2,1);

% Calculating optimal alpah using min and max eigenvalues of the 2D
% Laplacian matrix
minLamda = 4-4*cos(pi/(n+1));
maxLamda = 4-4*cos(n*pi/(n+1));
alpha = 2 /(minLamda+maxLamda);

% Creating initial conditions
x_initial = sparse(n^2,1);
tol = 1e-6;

% setting parameter to display results at every Mth iteration
M = 20;

% Calculating spectral radius of iteration matrix and appox number of iters
spectrad = (maxLamda-minLamda)/(minLamda+maxLamda);
fprintf('\n\n     Spectral radius of iteration matrix = %1.4f ',spectrad)
fprintf('\n     Approximate number of iterations = %4.0f \n\n\n',log10(tol)...
/log10(0.9595))


% Calling richardson routine
x = richardson(A,b,alpha,x_initial, tol,M);

% Checking that Richardson routine calculates the right answer
fprintf('\n     Inf-norm error between Richardson approx and backslash = %1.4e',norm(x-A\b,i


    function [A] = ConvectionDiffusion(beta,gamma,n)

        e = ones(n,1);

        % Creating sparse diagonal matrices
        I = spdiags(e,0,n,n);
        I1 =spdiags(e,1,n,n);
        I2 = spdiags(e,-1,n,n);

        % Creating 1D Convection-Diffusion matricies
        Abeta = 2*I +(beta-1)*I1 - (beta+1)*I2;
        Agamma = 2*I +(gamma-1)*I1 - (gamma+1)*I2;

        % Creating 2D Convection-Diffusion matrix
        A = kron(I,Abeta)+kron(Agamma,I);

    end
```

```
function A=lcd(beta,gamma,n)

    %
    % lcd.m: Laplacian-Convection-Diffusion, simple code to generate the matrices
    %        for assignment 1
    % Usage:
    %   - to generate the Laplacian (symmetric positive definite), call with
    %   beta=gamma=0: A=lcd(0,0,n);
    % - to generate a convection-diffusion matrix (nonsymmetric) call, e.g.,
    % with beta=0.5, gamma=0.6: A=lcd(0.5,0.6,n);
    %
    % Note: output matrix is of size n^2-by-n^2
    %

    ee=ones(n,1);
    a=4; b=-1-gamma; c=-1-beta; d=-1+beta; e=-1+gamma;
    t1=spdiags([c*ee,a*ee,d*ee],-1:1,n,n);
    t2=spdiags([b*ee,zeros(n,1),e*ee],-1:1,n,n);
    A=kron(speye(n),t1)+kron(t2,speye(n));
end

function y = MatVec(A,x)

    % finding sparsity partern of A
    [ii,jj,~] = find(A);

    y = spalloc(length(x),1,length(x));

    % looping through nonzero components of A to calculate the matrix
    % vector multiplication
    for i = 1:length(ii)
        y(ii(i)) = y(ii(i))+A(ii(i),jj(i))*x(jj(i));
    end

end

function x_out = richardson(A,b,alpha,x_initial, tol,M)

    % Creating fprintf logs
    titlelog = '\n\n\n        %4s   |     %5s       |    %5s \n';
    iterlog = '        %4i    |    %1.4e    |     %1.6f   \n';

    % Using sparse MatVec routine to calulation A*x_intial
    r = b-MatVec(A,x_initial);
    % Since x_inital is a vector of zeros then r_norm = b_norm but here
```

```
        % I have writen a general Richardson scheme
        r_norm = norm(r);
        b_norm = norm(b);
        ii = 1;
        xx = x_initial;
        fprintf(titlelog,'Iter','RelRes','Conver rate')
        fprintf('      ---------------------------------------------\n')

        fprintf(iterlog,0,r_norm/b_norm, 0)
        check = r_norm/b_norm;
        while check > tol
            % Storing old residual
            resold = r_norm/b_norm;
            % updating xx
            xx = xx +alpha*r;
            % Recalculating residual
            r = b-MatVec(A,xx);
            r_norm = norm(r);
            % printing results
            check = r_norm/b_norm;
            if mod(ii,M) == 0 || check < tol
                fprintf(iterlog,ii,r_norm/b_norm,1/(resold*b_norm/r_norm))
            end
            ii = ii + 1;
        end
        x_out = xx;
    end

end
```

```
    ===========================================================
      Question 1a) beta = 2.0000, gamma = -10.0000 and n =   4
    ===========================================================

      Inf-norm of A-Achen = 0.0000e+00

      Inf-norm error between MatVec and inbuild matrix-vector
                          product routine:

            Test  1: error = 0.0000e+00
            Test  2: error = 0.0000e+00
            Test  3: error = 0.0000e+00
            Test  4: error = 0.0000e+00
            Test  5: error = 0.0000e+00
            Test  6: error = 0.0000e+00
```

```
Test  7: error = 0.0000e+00
Test  8: error = 0.0000e+00
Test  9: error = 0.0000e+00
Test 10: error = 0.0000e+00
```

```
===========================================================
  Question 1b) beta =  0, gamma =  0 and n =  10
===========================================================

 Spectral radius of iteration matrix = 0.9595
 Approximate number of iterations =  334

    Iter   |     RelRes       |    Conver rate
  ---------------------------------------------
       0    |    1.0000e+00    |      0.000000
      20    |    3.3910e-01    |      0.959423
      40    |    1.4826e-01    |      0.959490
      60    |    6.4840e-02    |      0.959493
      80    |    2.8358e-02    |      0.959493
     100    |    1.2403e-02    |      0.959493
     120    |    5.4244e-03    |      0.959493
     140    |    2.3724e-03    |      0.959493
     160    |    1.0376e-03    |      0.959493
     180    |    4.5379e-04    |      0.959493
     200    |    1.9847e-04    |      0.959493
     220    |    8.6802e-05    |      0.959493
     240    |    3.7963e-05    |      0.959493
     260    |    1.6604e-05    |      0.959493
     280    |    7.2617e-06    |      0.959493
     300    |    3.1760e-06    |      0.959493
     320    |    1.3890e-06    |      0.959493
     328    |    9.9780e-07    |      0.959493

 Inf-norm error between Richardson approx and backslash = 6.1753e-06
```

We expect to see that the rate of convergence should be the same as the spectral radius of the iteration matrix. We can see that this is indeed the case from the table above. Using the formula

$$k \approx \frac{-m}{\log_{10}\rho},$$

where k, m and ρ are the number of iteration, $-\log_{10} tol$ and the spectral radius respectively we see that our approximation for the number of iterations is pretty accurate.