Table of Contents


```
function A1q3()

close all
clc

sigma = 1;
tau = 0;
n = 2^8;
h = 1/(n+1);

beta = sigma*h/2;
gamma = tau*h/2;
fprintf('\n    =====================================================')
fprintf('\n      Solving the Convection-Diffusion equation with\n')
fprintf('        beta = %2.4f, gamma = %2.4f and n = %3i\n',beta,gamma,n)
fprintf('    =====================================================\n')
% Creating discretised Convection Diffusion matrix
A = ConvectionDiffusion(beta,gamma,n);

% Defin exact solution and RHS functions
u = @(x,y) sin(pi*x).*cos(pi*y);
f = @(x,y,sigma,tau) 2*pi^2*u(x,y)+sigma*pi*cos(pi*x).*cos(pi*y) ...
    -tau*pi*sin(pi*x).*sin(pi*y);

% Creating domain
x = linspace(0,1,n+2);
[X,Y] = meshgrid(x,x);


% Defining RHS vector
F = f(X,Y,sigma,tau);
% Applying  Dirichlet boundary conditions
F(:,1) = sparse(n+2,1);
F(:,end) = sparse(n+2,1);
F = h^2*F(:);

% Defining exact solution
uE = u(X,Y);
uE = uE(:);
```

```
% Calculating error
tic
uA = A\F;
Dtime = toc;
fprintf('\n      Time using backslash = %4.4f',Dtime)
fprintf('\n      solution norm = %1.4e \n\n\n',norm(uE-uA,inf));

% Setting up initial guess
x = sparse((n+2)^2,1);

% Defining iterative parameters
max_iter = 2000;
tol = 1e-6;
```

```
=====================================================
  Solving the Convection-Diffusion equation with
     beta = 0.0019, gamma = 0.0000 and n = 256
=====================================================

  Time using backslash = 0.3428
  solution norm = 1.2617e-05
```

Jacobi

```
[~,rJ,iterJ] = Jacobi(max_iter,tol,x,A,F);
```

```
                  ------------------
                  |      Jacobi     |
                  ------------------
  Residual after error 2000 interations = 3.3423e-02
  Time to run Jacobi = 3.7176
```

BLKJacobi

```
[~,rBLK,iterBLK] = BLKJacobi(max_iter,tol,x,A,F);
```

```
                  ------------------
                  |   BLK Jacobi    |
                  ------------------
  Residual after error 2000 interations = 2.8750e-02
  Time to run Jacobi = 8.3165
```

GS

```
[~,rGS,iterGS] = GS(max_iter,tol,x,A,F);
```

```
                  ------------------
```

```
                    |  Gauss-Siedel  |
                    ------------------
Residual after error 2000 interations = 2.8793e-02
Time to run GS = 5.5019
```

SOR

```
fprintf('                      ------------------     \n');
fprintf('                      |       SOR        |   \n');
fprintf('                      ------------------     \n');
ii = 0;
rSOR = cell(9,1);
iterSOR = sparse(9,1);
test = zeros(9,1);
omega = zeros(9,1);
for i = 1.1:.1:1.9
    ii = ii + 1;
    [~,rSORR,iterSOR(ii)] = SOR(max_iter,tol,x,A,F,i);
    rSOR{ii} = rSORR;
    test(ii) = rSORR(end);
    omega(ii) = i;
end
[~,index]=min(test);
```

```
                    ------------------
                    |      SOR       |
                    ------------------
Omega = 1.1
Residual after error 2000 interations = 2.6957e-02
Time to run SOR = 6.0345

Omega = 1.2
Residual after error 2000 interations = 2.4834e-02
Time to run SOR = 6.0342

Omega = 1.3
Residual after error 2000 interations = 2.2359e-02
Time to run SOR = 6.1162

Omega = 1.4
Residual after error 2000 interations = 1.9451e-02
Time to run SOR = 6.0760

Omega = 1.5
Residual after error 2000 interations = 1.6023e-02
Time to run SOR = 6.0713

Omega = 1.6
Residual after error 2000 interations = 1.2006e-02
Time to run SOR = 6.0669

Omega = 1.7
```

```
Residual after error 2000 interations = 7.4615e-03
Time to run SOR = 6.0467

Omega = 1.8
Residual after error 2000 interations = 2.9298e-03
Time to run SOR = 6.0238

Omega = 1.9
Residual after error 2000 interations = 2.0813e-04
Time to run SOR = 5.9946
```

GMRES

```
tic
[~,~,~,iterGM,rGM] = gmres(A,F,[],tol,max_iter);
time = toc;
% Produces results
fprintf('                       -----------------     \n');
fprintf('                       |     GMRES     |   \n');
fprintf('                       -----------------     \n');
fprintf('      Residual error after %4.0f interations = %1.4e \n'...
    ,iterGM(end),rGM(iterGM(end)))
fprintf('      Time to run CG = %2.4f\n\n',time)
```

```
                 -----------------
                 |     GMRES     |
                 -----------------
Residual error after  360 interations = 4.3652e-08
Time to run CG = 67.4629
```

PGMRES

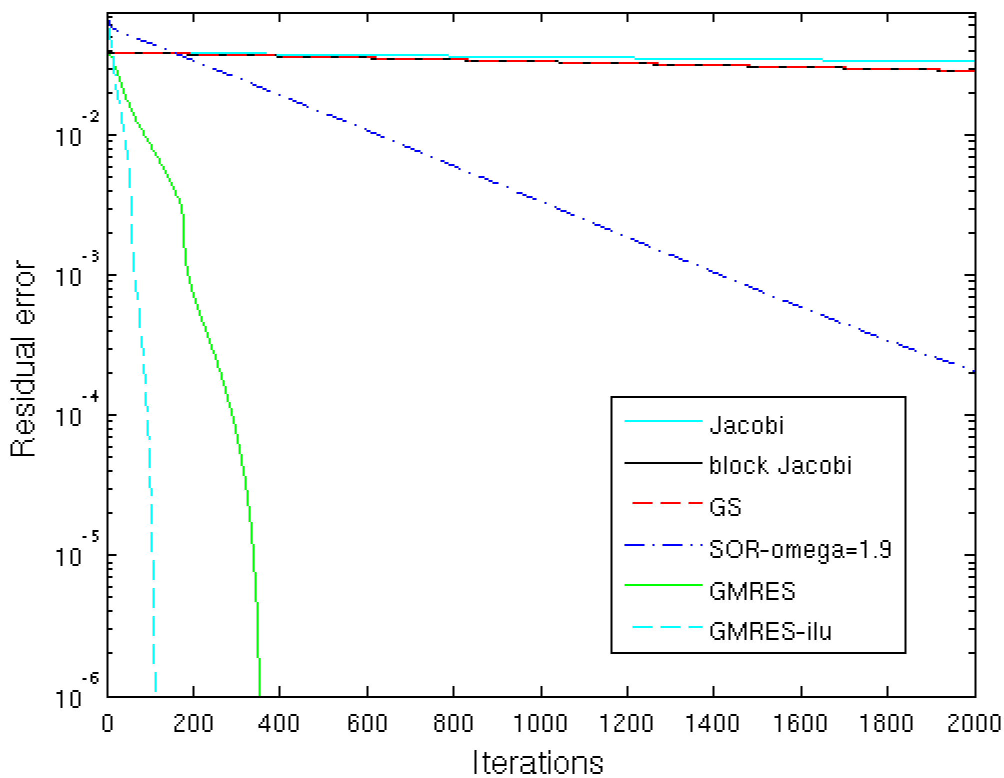
```
tic
setup.type = 'nofill';
[L,U] = ilu(A,setup);
[~,~,~,iterPGM,rPGM] = gmres(A,F,[],tol,max_iter,L,U);
time = toc;
% Produces results
fprintf('                       -----------------     \n');
fprintf('                       |    PGMRES     |   \n');
fprintf('                       -----------------     \n');
fprintf('      Residual error after %4.0f interations = %1.4e \n'...
    ,iterPGM(end),rPGM(iterPGM(end)))
fprintf('      Time to run CG = %2.4f\n\n',time)

rSOR = rSOR{index};
m_iter = max([iterJ,iterBLK,iterGS,iterSOR(index),iterGM(end),iterPGM(end)]);
m_res = max([rJ(1),rBLK(1),rGS(1),rSOR(1),rGM(1),rPGM(1)]);
```

```
                 -----------------
                 |    PGMRES     |
```

```
                    -----------------
Residual error after  198 interations = 6.7226e-08
Time to run CG = 21.6125
```

Plotting commands

```
semilogy(rJ,'c'); hold on
semilogy(rBLK,'k'); hold on
semilogy(rGS,'r--');hold on
semilogy(rSOR,'b-.');hold on
semilogy(rGM,'g');hold on
semilogy(rPGM,'c--');hold on
axis([0 m_iter tol m_res]);
legend('Jacobi','block Jacobi','GS',['SOR-omega=',num2str(omega(index))]...
    ,'GMRES','GMRES-ilu','Location','Best')
xlabel('Iterations','fontsize',14);
ylabel('Residual error','fontsize',14);
```




```
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;

    % applying boundary conditions
    AbetaBC = [1 sparse(1,n+1);[-(beta+1); sparse(n-1,1)] ...
        Abeta [sparse(n-1,1); beta-1];sparse(1,n+1) 1];
    AgammaBC = [2 -2 sparse(1,n);[-(gamma+1); sparse(n-1,1)]...
        Agamma [sparse(n-1,1); gamma-1];sparse(1,n) -2 2 ];

    % Creating 2D Convection-Diffusion matrix
    A = kron(AbetaBC,speye(n+2))+kron(speye(n+2),AgammaBC);

end


function [x,rJ,i] = Jacobi(max_iter,tol,x,A,b)
    % Jacobi method
    tic
    % Creating vector of diagonals of A
    D = spdiags(A,0);
    b_norm = norm(b);
    r = b;

    % Carring out Jacobi
    for i = 1:max_iter

        % updating x
        x = x + r./D;
        r = b-A*x;
        rJ(i) = norm(r);

        if (rJ(i)/b_norm) < tol
            break
        end
    end
    time = toc;
    % Produces results
    fprintf('                      ------------------     \n');
    fprintf('                      |     Jacobi     |   \n');
    fprintf('                      ------------------     \n');
    fprintf('      Residual after error %4.0f interations = %1.4e \n'...
        ,i,rJ(i))
    fprintf('      Time to run Jacobi = %2.4f\n\n',time)
end


function [x,rJ,i] = BLKJacobi(max_iter,tol,x,A,b)
    % Jacobi method
    tic
    % Creating Block Jacobi
    BLKjac = spdiags([spdiags(A,-1),spdiags(A,0),spdiags(A,1)]...
```

```
        ,[-1:1],(n+2)^2,(n+2)^2);
    b_norm = norm(b);
    r = b;
    % Factoring BLKjac so can use backwards and forwards subsitution at
    % each step
    [LowT,UpT,Piv] = lu(BLKjac,'vector');
    % Carring out BLKJacobi
    for i = 1:max_iter

        % updating x
        x = x + UpT\(LowT\(r(Piv,:)));

        r = b-A*x;
        rJ(i) = norm(r);

        if (rJ(i)/b_norm) < tol
            break
        end
    end
    time = toc;
    % Produces results
    fprintf('                              ------------------     \n');
    fprintf('                              |   BLK Jacobi    |    \n');
    fprintf('                              ------------------     \n');
    fprintf('      Residual after error %4.0f interations = %1.4e \n'...
        ,i,rJ(i))
    fprintf('      Time to run Jacobi = %2.4f\n\n',time)
end

function [x,rGS,i] = GS(max_iter,tol,x,A,b)
    tic
    % Creating lower triangular
    E = tril(A);
    b_norm = norm(b);
    r = b;

    % Carring out GS
    for i = 1:max_iter

        % updating x
        x = x + E\r;
        r = b-A*x;
        rGS(i) = norm(r);

        if (rGS(i)/b_norm) < tol
            break
        end
    end
    time = toc;
    % Produces results
    fprintf('                              ------------------     \n');
    fprintf('                              |  Gauss-Siedel   |    \n');
    fprintf('                              ------------------     \n');
    fprintf('      Residual after error %4.0f interations = %1.4e \n'...
```

```
            ,i,rGS(i))
        fprintf('      Time to run GS = %2.4f\n\n',time)
    end


    function [x,rSOR,i] = SOR(max_iter,tol,x,A,b,omega)
        tic
        % Creating lower triangular
        E = tril(A);
        [N,~] = size(A);
        % Creating sparse diagonal matrix
        D = spdiags(spdiags(A,0),0,N,N);

        % Calculating the matrix which we need to solve at each step
        S = ((1/omega-1)*D+E);
        b_norm = norm(b);
        r = b;

        % Carring out SOR
        for i = 1:max_iter

            % updating x
            x = x + S\r;
            r = b-A*x;
            rSOR(i) = norm(r);

            if (rSOR(i)/b_norm) < tol
                break
            end
        end
        time = toc;
        % Produces results
        fprintf('      Omega = %1.1f \n',omega)
        fprintf('      Residual after error %4.0f interations = %1.4e \n'...
            ,i,rSOR(i))
        fprintf('      Time to run SOR = %2.4f\n\n',time)
    end

end
```

Published with MATLAB® 7.14