close all

% Number of points

numpoint = 10;

%..........Velocity

Pe = 7;

% Diffusion coefficient

diffcoef = 1;

% Domain length

lengthdom = 1;

% element number

numelem = numpoint - 1;

% element length

elelength = lengthdom/numelem;

% Calculation of velocity

Velocity = (2\*Pe\*diffcoef)/lengthdom;

timearray = 0:elelength:lengthdom;

x=0:0.01:1;

% Analytical solution

phi\_exact = zeros(length(x),1);

for i=1:length(x)

    phi\_exact(i) = (exp(Velocity\*x(i)/diffcoef)-exp(Velocity\*lengthdom/diffcoef))/(1-exp(Velocity\*lengthdom/diffcoef));

end

plot(x,phi\_exact,'-k','LineWidth',1.2);

hold on

grid

% Standard galerkin method

a0 = 0; % tau = 0

% Petrov galerkin method

a1 = 1/(0.5\*Pe); % tau alpha equals to alpha critical

a2 = 1-(1/abs(Pe)); % alpha smaller than alpha critical

a3 = 2/(0.5\*Pe); % tau alpha equals alpha critical

a4 = coth(abs(Pe))-1/abs(Pe); % tau equals alpha optimal

alpha = [a0 a1 a2 a3 a4];

% alpha = [a1 a2 a3 a4 ]; % tau alpha = [alpha]

for tau=1:length(alpha)

    % Node connectivity

    lpoints = zeros(numelem,2);

    for i=1:(numelem)

        lpoints(i,1) = i;

        lpoints(i,2) = i+1;

    end

    % Connection matrix

    Convmat = zeros(numpoint,numpoint);

% Diffusion matrix

Diffmat = zeros(numpoint,numpoint);

% Extra diffusion matrix for petrov galerkin methods

extraDiff = zeros(numpoint,numpoint);

for elem = 1:numelem

    % current element convection matrix

    currentelem = Velocity\*[1 -1; -1 1];

    % current element diffusion matrix

    currentdiff = diffcoef/elelength \* [1 -1; -1 1];

    % current element extra diffusion matrix

    extraDiff(alpha) = alpha(tau) \* diffcoef/elelength \* [1 -1; -1 1];

    % global nodes of element

    points = lpoints(elem,:);

    % Matrix

    Convmat(points,points) = Convmat(points,points)+currentelem;

    Diffmat(points,points) = Diffmat(points,points)+currentdiff;

    extraDiff(points,points) = extraDiff(points,points)+currentextradiff;

end

phi = zeros(numpoint,1);

% Boundary Condition

phi(1)=1; % boundary condition at entry phi = 1 at inlet

phi(end)=0; % boundary condition at exit phi = 0 at outlet

% Matrix created:

A = Convmat+Diffmat+extraDiff;

% Code continued on the next cell

% continued from previous cell

% Red = A(2:numpoint-1,2:numpoint-1);

rhs\_red = phi(2:numpoint-1);

% Reduced system

newphi=A\phi;

phi(2:numpoint-1)=newphi(2:numpoint-1);

% Plot

plot(timearray,phi,'Marker','o','LineWidth',0.8,'MarkerSize',10);

end

legend('Analytical Solution','PetrovGalerkin a = 1','a < alpha critical','a = alpha optimal','location','north west');

xlabel('Domain Length (x)')

ylabel('Scalar Quantity (\phi)')

title(['Convection Diffusion for Peclet number(Pe) = ',num2str(Pe)],'FontSize',12,'FontWeight','normal','Color','black');