Numerical Methods 4H Assignment 2 - A two-point boundary value problem

1107023m

December 2014

0.1 Question 1

We use a uniform partition of [0,1] by (N + 1) evenly spaced nodes
$0 = x_0 < x_1 < \cdots < x_n = 1$ where $x_i = i/N$ then $h = 1/N$ and $x_i = hi$.
Let u_i denote $u(x_i)$

0.2 Question 2

At each interior point of [0,1] we have the following formulas:

$$u''(x_i) = \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + O(h^2) \tag{1}$$

$$u'(x_i) = \frac{u_{i+1} - u_{i-1}}{2h} + O(h^2) \tag{2}$$

The finite difference approximation for $u(x)$ is now obtained by enforcing the original equation at each interior node of [0,1] that is:

$$\begin{aligned} &\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + exp(u_i)\frac{u_{i+1} - u_{i-1}}{2h} = \mu sin(2\pi x_i) \\ &\iff 2u_{i+1} - 4u_i + 2u_{i-1} + hexp(u_i)(u_{i+1} - u_{i-1}) = 2h^2\mu sin(2\pi x_i)0 \\ &\iff u_{i+1}(2 + hexp(u_i)) - 4u_i + u_{i-1}(2 - hexp(u_i)) = 2h^2\mu sin(2\pi x_i) \end{aligned} \tag{3}$$

for $i \in [1, N-1]$

0.3 Question 3

The $O(h^2)$ accurate backwards finite difference formula for u' is:

$$\frac{u_{i-2} - 4u_{i-1} + 3u_i}{2h} = u'(x_i) + O(h^2) \tag{4}$$

Substitute this into the boundary equation $u'(1) + u^3(1) = 0$:

$$\begin{aligned} &u'(1) + u^3(1) = 0 \\ &\iff u'(x_N) + u_N^3 = 0 \\ &\iff \frac{u_{N-2} - 4u_{N-1} + 3u_N}{2h} + u_N^3 = 0 \\ &\implies u_N(2hu_N^2 + 3) - 4u_{N-1} + u_{N-2} = 0 \end{aligned} \tag{5}$$

which is the difference equation for node x_N

The difference equation for node x_0 is simply $u_0 = 1$

0.4 Question 4

Let F(u) = 0 be a system of N equations where

$$\begin{aligned}
F_1 &= u_2(2+hexp(u_1)) - 4u_1 + (2-hexp(u_1)) - 2h^2\mu sin(2\pi x_1) = 0 \\
\vdots &= \quad \vdots \\
F_i &= u_{i+1}(2+hexp(u_i)) - 4u_i + u_{i-1}(2-hexp(u_i)) - 2h^2\mu sin(2\pi x_i) = 0 \\
\vdots &= \quad \vdots \\
F_{N-1} &= u_N(2+hexp(u_{N-1})) - 4u_{N-1} + u_{N-2}(2-hexp(u_{N-1})) - 2h^2\mu sin(2\pi x_{N-1}) = 0 \\
F_N &= u_N(2hu_N^2+3) - 4u_{N-1} + U_{N-2} = 0
\end{aligned} \tag{6}$$

Then calculating the Jacobian matrix J where entry $J_{ij} = \frac{\partial F_i}{\partial u_j}$:

Define the following equations for i $\in [2, n-1]$:

$L_i = \frac{\partial F_i}{\partial u_{i-1}} = 2 - hexp(u_i)$
$U_i = \frac{\partial F_i}{\partial u_{i+1}} = 2 + hexp(u_i)$
$D_i = \frac{\partial F_i}{\partial u_i} = (u_{i+1} - u_{i-1})hexp(u_i) - 4$

$$J = \begin{bmatrix}
(u_2-1)hexp(u_1)-4 & 2+hexp(u_1) & 0 & 0 & \dots & \dots & \dots & 0 \\
L_2 & D_2 & U_2 & 0 & \dots & \dots & \dots & 0 \\
0 & L_3 & D_3 & U_3 & 0 & \dots & \dots & \dots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \dots & \dots \\
\vdots & \vdots & \ddots & L_i & D_i & U_i & \dots & \dots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \dots & \dots \\
0 & \dots & \dots & \dots & 0 & L_{n-1} & D_{n-1} & U_{n-1} \\
0 & \dots & \dots & \dots & \dots & 1 & -4 & 6hu_n^2+3
\end{bmatrix}$$

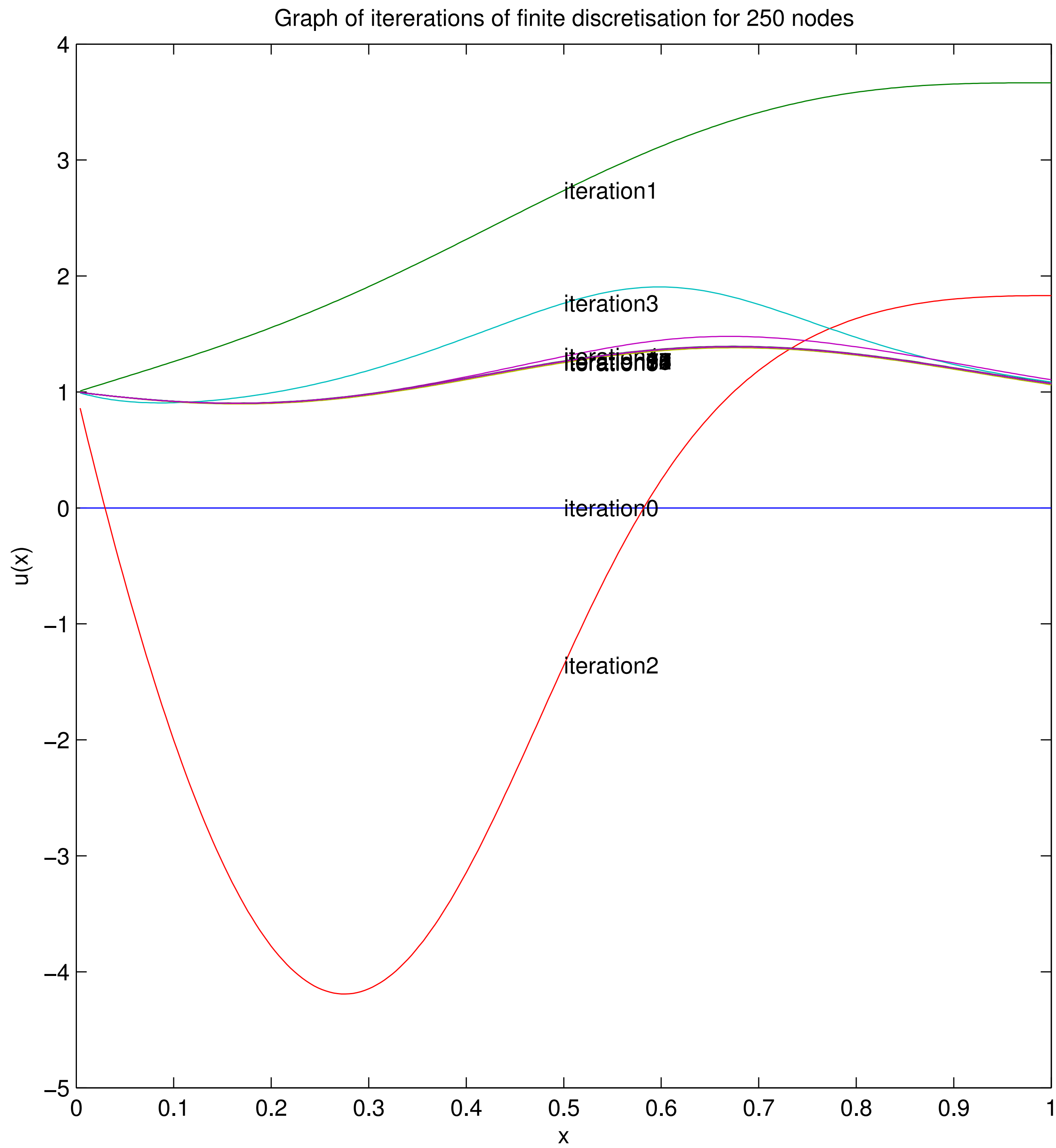
0.5 Question 5

See file q5.m for the same code as below:

```
% Code to calculate the solution to a
% nonlinear BVP with Dirichlet and Robin BCs

% Some physical parameter.
mu = 10;

% Define the interval over which solution is calculated.
a= 0; b= 1;

% Define N.
N= 250;

% Define the grid spacing.
h=(b-a)/N;

% Define the grid. Note that there is no need to solve at x=a.
x = reshape(linspace(a+h,b,N),N,1);

% Define an initial guess for the solution
% of the BVP (make sure it is a column vector)
U= zeros(N, 1);

% Define a tolerance for the termination of Newton-Raphson.
tol= 10^(-8);
% Ensure that F is such that at least one iteration is done.
F=ones(N,1);
J=zeros(N,N);

% Store the initial guess in SOL.
SOL= U;

% Loop while the norm(F) is greater than tol.
while (norm(F)>tol )

    %% Define F(u).
    % Boundary conditions.
     F(N) = (U(N) * ((2 * h * (U(N)^2)) + 3)) - (4 * (U(N-1))) + U(N-2);

    % Finite difference approximation to ODE at interior nodes.
    F(1) = (U(2) * (2 + h * exp(U(1)))) - (4 * U(1)) + (2 - (h * exp(U(1))));
    F(1) = F(1) - (2 * (h^2) * mu * sin(2 * pi * x(1)));
    for i=2:N-1
        F(i)= (U(i+1) * (2 + h * exp(U(i)))) - (4 * U(i)) + U(i-1)*(2 - h * exp(U(i)));
        F(i) = F(i) - (2 * (h^2) * mu * sin(2 * pi * x(i)));
    end;

    %% Define the Jacobian J.
    % First row corresponds to BC at x=0.
    J(1,:) = horzcat(((U(2) - 1) * h * exp(U(1))) - 4, 2 + (h * exp(U(1))), zeros(1, N-2));

    % Last row corresponds to BC at x=1.
    %J(N,:) = horzcat(zeros(1, N-2),  2/(h^2), -2/(h^2) - (U(N)^2) * exp(U(N))*(U(N) + 3));
    J(N,:) = horzcat(zeros(1, N-3), 1, -4, (4 * h * U(N)^2) + 3);

    % Intermediate rows correspond to F(i)=...
    for i=2:(N-1)
        % Diagonal entries.
        J(i,i)= ((U(i+1) - U(i-1)) * h * exp(U(i))) - 4;
        % Left diagonal entries
        J(i, i-1) = 2 - (h * exp(U(i)));
        % Right diagonal entries
        J(i, i+1) = 2 + (h * exp(U(i)));
    end

    %% Having defined F and J, update the approximate solution to
    % the difference equations.

    U=U-J\F;

    %% Store the new approximation.
```

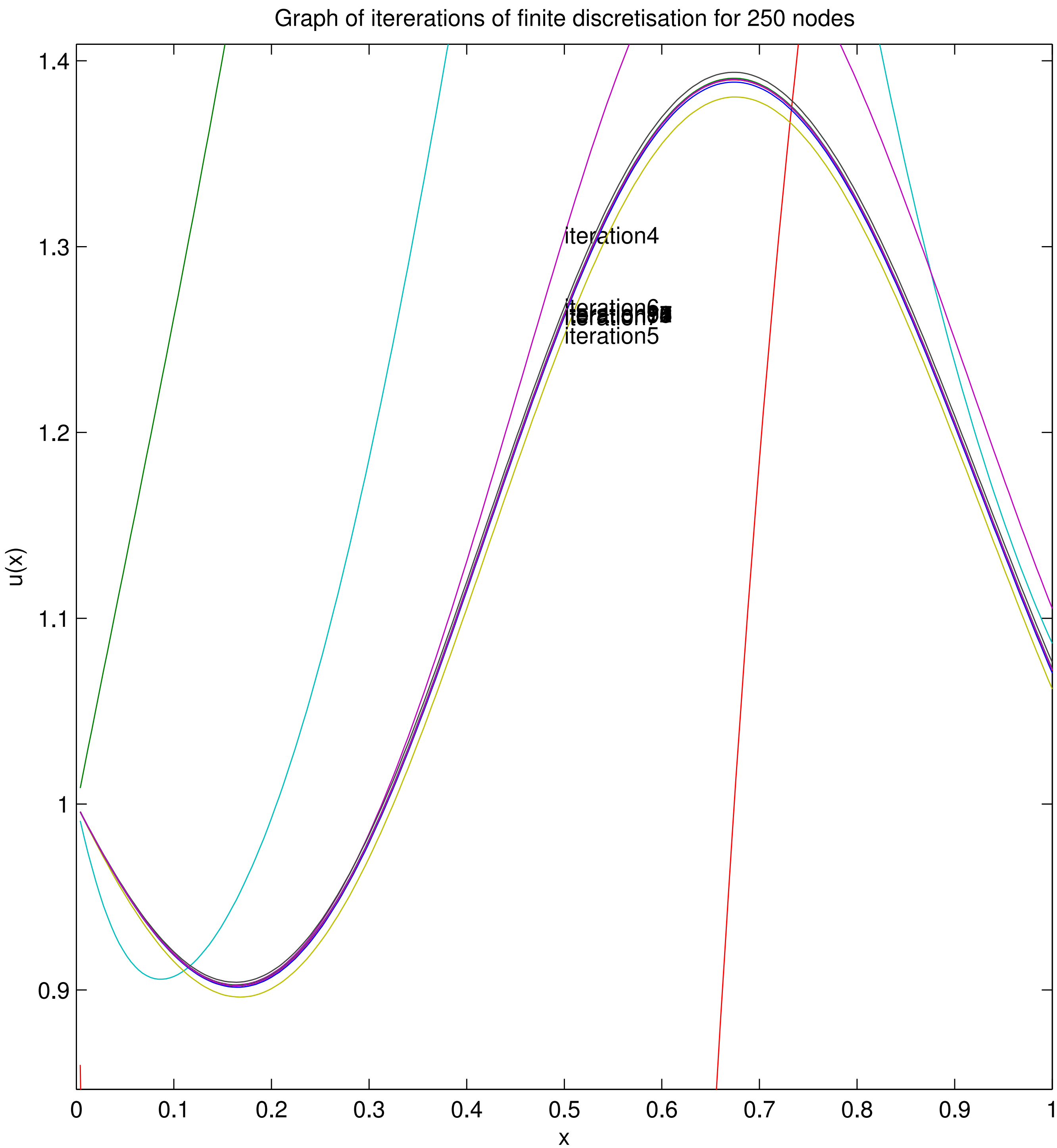
```
    SOL= [SOL,U];
    size(SOL)
end

%% Insert your plot commands here.
plots = []
labels = []
sz = size(SOL);
for i  = 1:(sz(2))
        p = plot(x, SOL(:,i));
        text(0.5, SOL(N/2,i), strcat('iteration  ', int2str(i - 1)));
        plots = [plots, p];
        hold all;
end
title('Graph of itererations of finite discretisation for 250 nodes')
xlabel('x'); % x-axis label
ylabel('u(x)'); % y-axis label
hold off
```

Plotting each iteration we get the following graph:



After around iteration 3 we can clearly see a convergence of values.

Zooming in vertically:



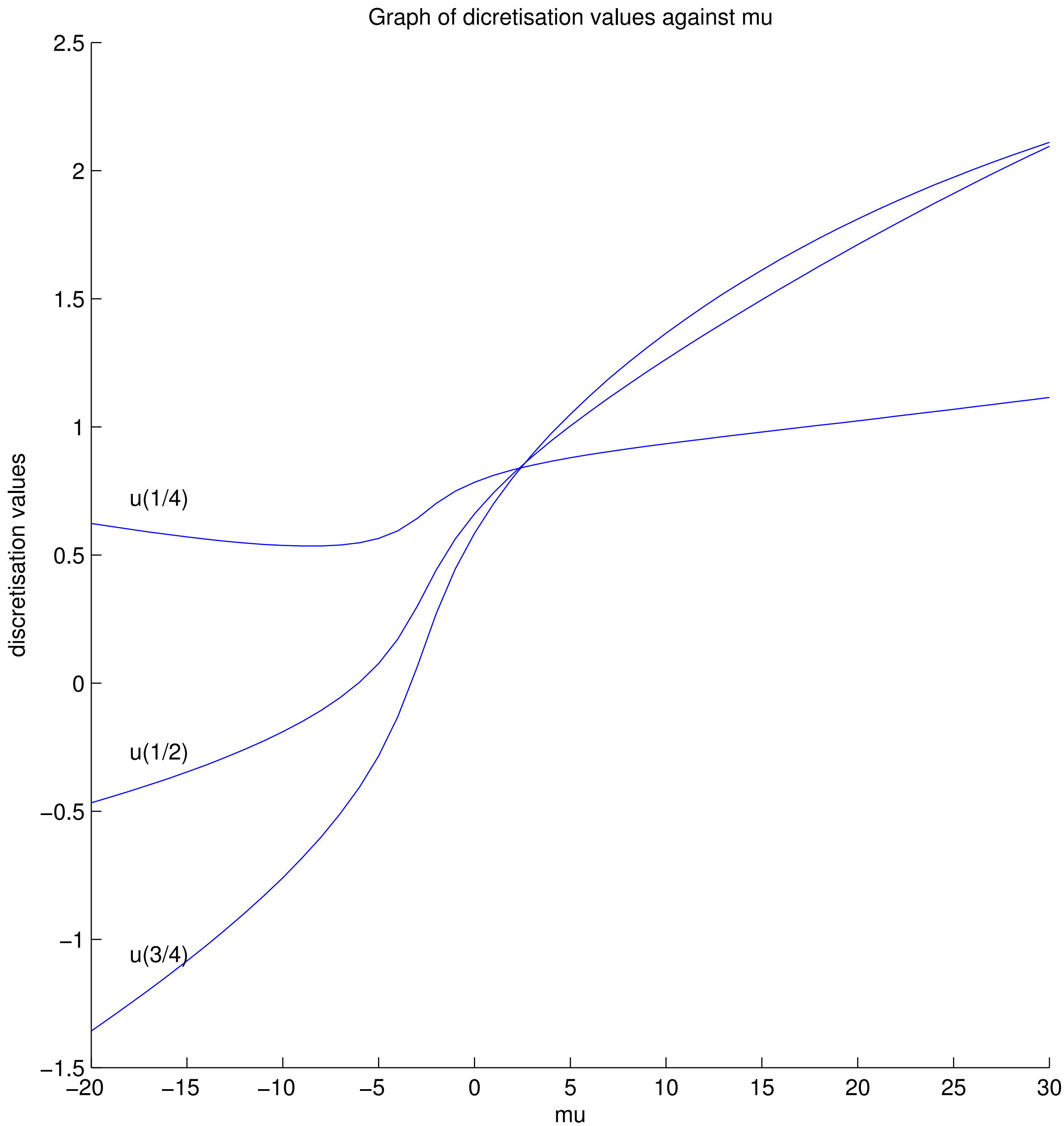
The convergeance can be seen more clearly as well as the Newton-Raphson iterations as all values in iteration 4 are larger than those in iteration 5 and all values in further iterations are inbetween these. Zooming in vertically on any further iterations doesn't really give much information visually without manually scrolling.

0.6 Question 6

Modified question 5 matlab code as a function. See file bvp.m for the same code as below:

```
function res = bvp(mu)
% Code to calculate the solution to a
% nonlinear BVP with Dirichlet and Robin BCs

% Define the interval over which solution is calculated.
a= 0; b= 1;

% Define N.
N = 100;

% Define the grid spacing.
h=(b-a)/N;

% Define the grid. Note that there is no need to solve at x=a.
x = reshape(linspace(a+h,b,N),N,1);

% Define an initial guess for the solution
% of the BVP (make sure it is a column vector)
U= ones(N, 1);

% Define a tolerance for the termination of Newton-Raphson.
tol= 10^(-8);

% Ensure that F is such that at least one iteration is done.
F=ones(N,1);
J=zeros(N,N);

% Loop while the norm(F) is greater than tol.
while (norm(F)>tol )

    %% Define F(u).
    % Boundary conditions.
    F(N) = (U(N) * ((2 * h * (U(N)^2)) + 3)) - (4 * (U(N-1))) + U(N-2);

    % Finite difference approximation to ODE at interior nodes.
    F(1) = (U(2) * (2 + h * exp(U(1)))) - (4 * U(1)) + (2 - (h * exp(U(1))));
    F(1) = F(1) - (2 * (h^2) * mu * sin(2 * pi * x(1)));

    for i=2:N-1
        F(i)= (U(i+1) * (2 + h * exp(U(i)))) - (4 * U(i)) + U(i-1)*(2 - h * exp(U(i)));
        F(i) = F(i) - (2 * (h^2) * mu * sin(2 * pi * x(i)));
    end;

    %% Define the Jacobian J.
    % First row corresponds to BC at x=0.
    J(1,:) = horzcat(((U(2) - 1) * h * exp(U(1))) - 4, 2 + (h * exp(U(1))), zeros(1, N-2));

    % Last row corresponds to BC at x=1.
    J(N,:) = horzcat(zeros(1, N-3), 1, -4, (4 * h * U(N)^2) + 3);

    % Intermediate rows correspond to F(i)=...
    for i=2:(N-1)
        % Diagonal entries.
        J(i,i)= ((U(i+1) - U(i-1)) * h * exp(U(i))) - 4;
        % Left diagonal entries
        J(i, i-1) = 2 - (h * exp(U(i)));
        % Right diagonal entries
        J(i, i+1) = 2 + (h * exp(U(i)));
    end

    %% Having defined F and J, update the approximate solution to
    % the difference equations.
    U=U-J\F;

end

% Return values of U at X = 1/4, X = 1/2, and X = 3/4
res = [U(N/4), U(N/2), U((3*N)/4)];

end
```

See q6.m for matlab code which generates this graph:



0.7 Question 7

See q7.m for matlab code which generates this graph: