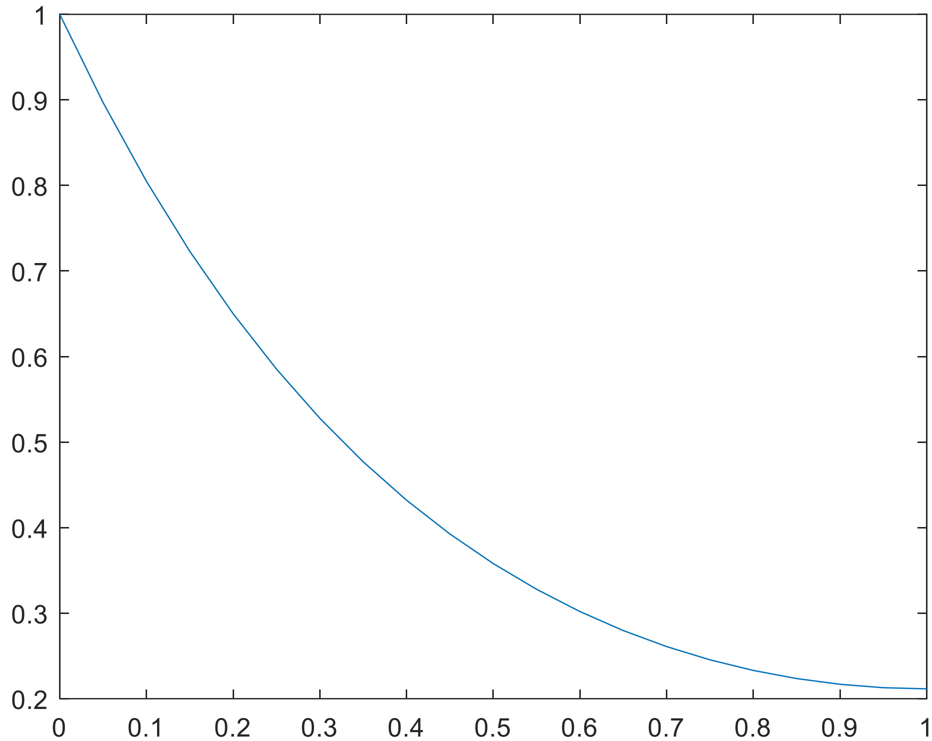
```matlab
function Question1
clc;close all;                          % Clear workspace and command window,
                                        %      close all figures

L=1;                                    % Set the length of the spatial domain
n=21;                                   % Set the number of node points
dx=L/(n-1);                             % Calculate the separation of node pts
x=linspace(0,n-1,n)*dx;                 % Set the locations of the node points
itr_max=1500;                           % Set the maximum iterations
tol=1e-6;                               % Specify the convergence criterion
alpha=1.8;                              % Set the overrelaxation factor
                                        % Initialize unknown values of the
                                        %      dependent variables
y0=30*ones(n,1);                        % Set the initial guesses for the node
y0(1)=1;                                %     point values
y0(end)=(4*y0(n-1)-y0(n-2))/3;
Y=y0;                                   % Use initial guesses for initial
Y_c=Y;                                  %      value of Y
for i=1:itr_max
    error=0;
    for j=2:n
        if j==n
            Y_c(j)=(4*Y_c(n-1)-Y_c(n-2))/3;
        else
            Y_c(j)=get_Y_c(Y,j);
        end
                % Using nested function calculate Y_cal
                                       % Apply SOR

        Y(j)=Y(j)+alpha*(Y_c(j)-Y(j));
                                       % Calculate relative error
        error_t=abs((Y_c(j)-Y(j))/Y_c(j));
                                       % Find maximum relative error
        if error_t>error; error=error_t; end
    end

    if error<tol; break; end           % Check for convergence
end
fprintf('Y at x=l: %1.3f\n', Y(end))

plot(x,Y)
%
% Nested function for applying recursion relation
%
    function yc=get_Y_c(YY,ii)
                                       % Apply recursion relation
        yc=(YY(ii-1)+YY(ii+1))/(5*dx^2+2);
    end
end
```

OUTPUT:

```
Y at x=l: 0.212
```

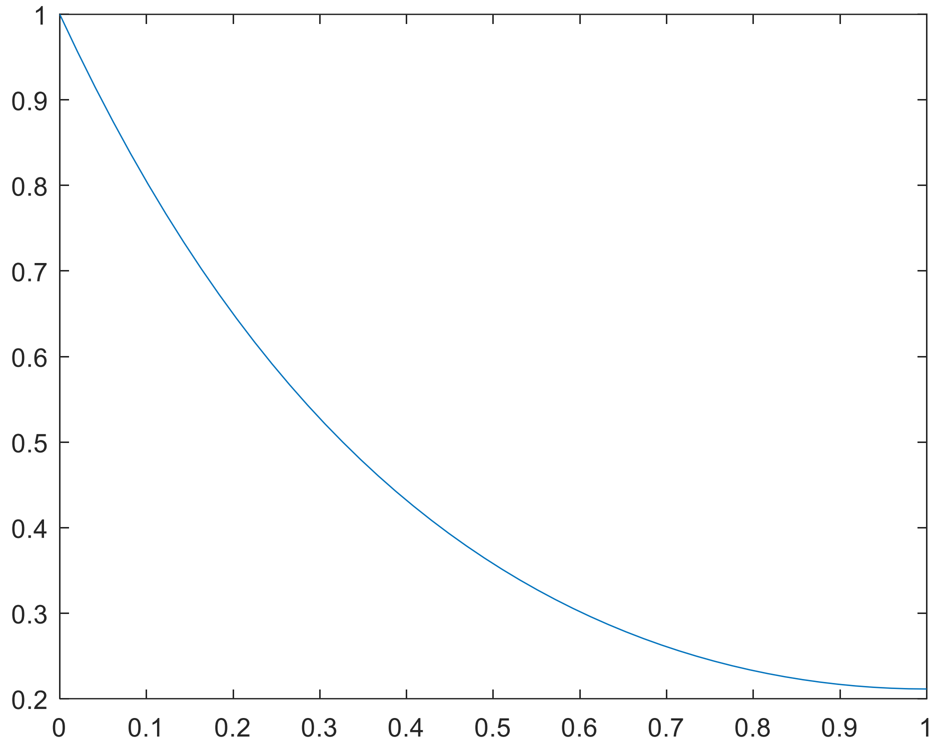


QUESTION 2

```
function Question2
    xx=linspace(0,1,50);
    solninit=bvpinit(xx,@funinit);
    soln=bvp4c(@odefun,@funbc,solninit);
    yy=deval(soln,xx,1);
    fprintf('Y at x=l: %1.3f\n', yy(end))
    plot(xx,yy)

    function dydx=odefun(x,y)
        dydx(1,1)=y(2);
        dydx(2,1)=5*y(1);
    end

    function residual=funbc(ya,yb)
        residual(1,1)=ya(1)-1; % y at 0 is 1
        residual(2,1)=yb(2); % y' at 1 is 0
    end

    function yinit=funinit(x)
        % Guess that y = 0 and y' = 1
        yinit=[0,1];
    end
end
```

OUTPUT:

```
Y at x=l: 0.211
```

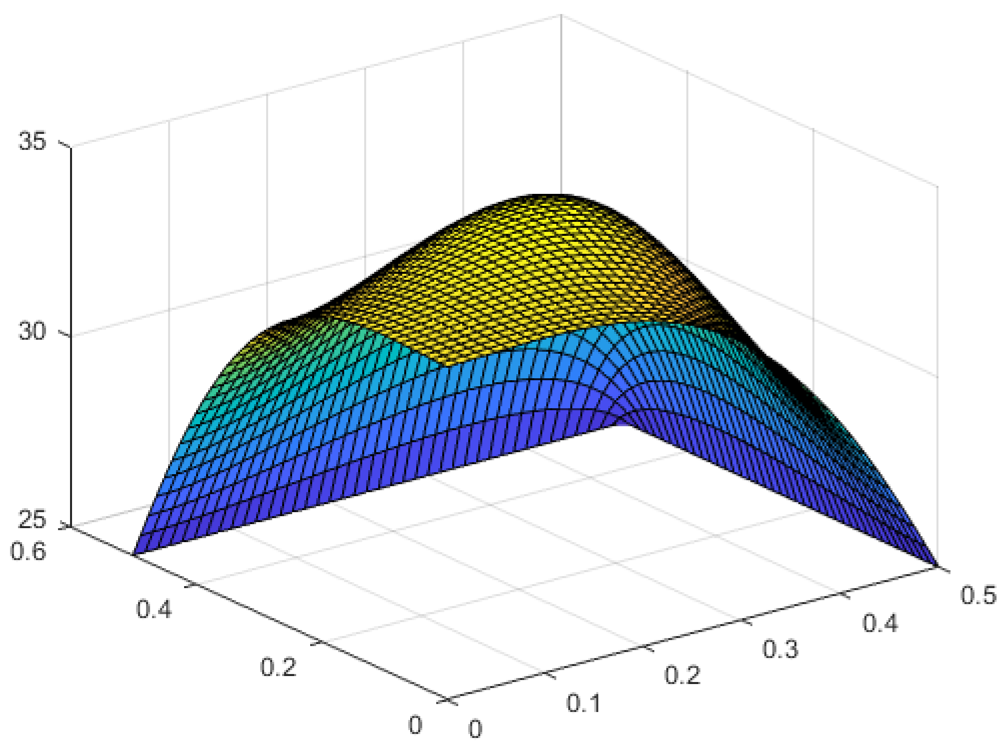


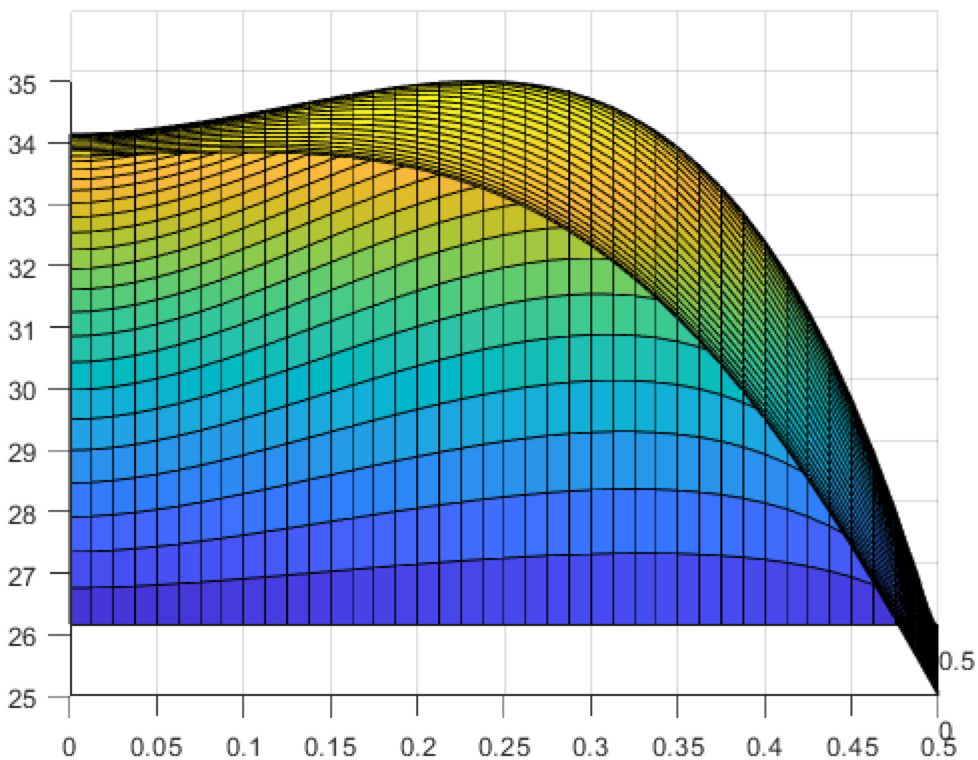
QUESTION 3:

```
function Question3
clear all;clc;close all;            % Clear workspace and command window,
                                    %      close all figures

L=0.5; y=L; n=41; dx=L/(n-1);       % Specify parameters of the problem
itr_max=1500; tol=10^-4; alpha=1.8;  % Set numerical parameters
T=zeros(n,n);                       % Set all element of T equal to zero
                                    % Set constant valued boundary values

T(n,:)=25*ones(1,n);
T(:,n)=25*ones(n,1);
T_c=T;
for k=1:itr_max                     % Iterate until itr_max or convergence
    error_max=0;                    % Set max error =0 at beginning of
                                    %      each iteration

    for i=2:n-1                     % Increment row numbers
         for j=2:n-1                % Increment column numbers
                                    % Apply recursion relation
             T_c=get_T_c(T,i,j);
                                    % Apply relaxation factor
             T(i,j)=T(i,j)+alpha*(T_c-T(i,j));
                                    % Calculate relative error for each
                                    %      node
             error_t=abs((T_c-T(i,j))/T_c);
                                    % Determine maximum relative error
             if error_t > error_max; error_max=error_t; end;
         end
    end
```

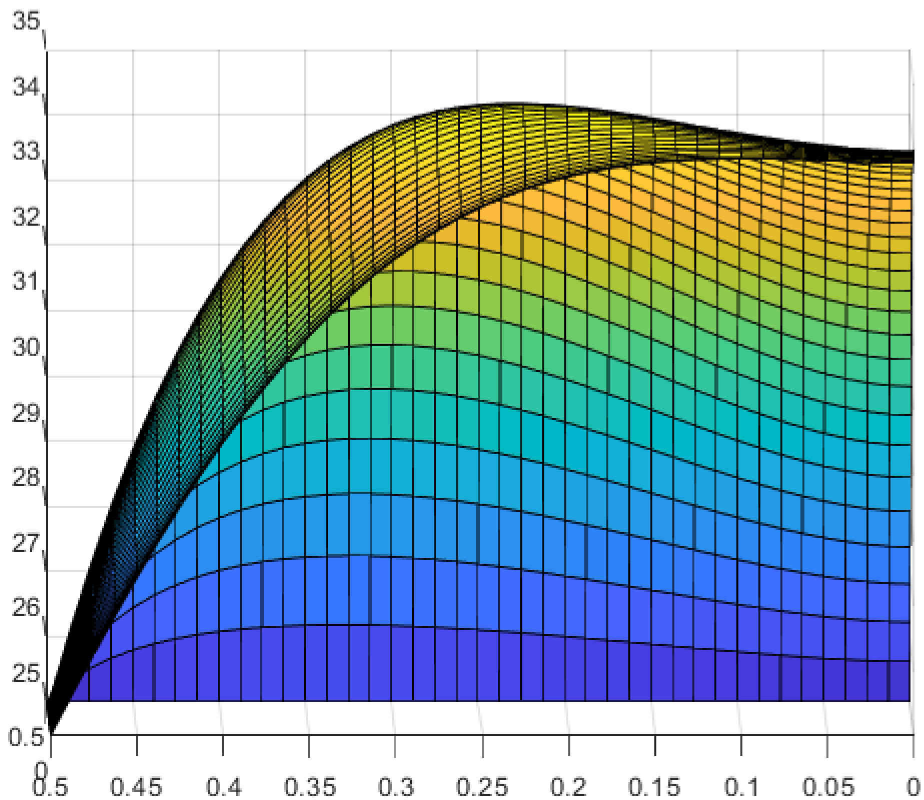
```
    for i=2:n-1; T(i,1)=(4*T(i,2)-T(i,3))/3; end
    for j=2:n-1; T(1,j)=(4*T(2,j)-T(3,j))/3; end
    T(1,1)=(T(2,1)+T(1,2))/2;
    if error_max < tol; break; end   % Convergence check
end
if k>itr_max-1                         % Check for maximum iteration limit
    fprintf('Maximum iterations exceeded without convergence\n')
end
x=linspace(0,0.5,n);                   % Set x values for contour map
y=linspace(0,0.5,n);                   % Set y values for contour map
[X,Y]=meshgrid(x,y);
% Be careful with the x, y coordinate.
% The 1st and 2nd indice represent row and colunm. However, it does not directly fit
to Cartesian
% coordinate. Need to flip "X" and "Y" in the countour plot
figure, contour(Y,X,T,8,'ShowText','on')                        % Generate contour map
from solution
figure, surf(Y,X,T)
%
% Nested function that applies the recursion relation
%
    function tc=get_T_c(TT,ii,jj)
        xx=dx*(ii-1); yy=dx*(jj-1);
        tc=(TT(ii+1,jj)+TT(ii-1,jj)+TT(ii,jj+1)+TT(ii,jj-1))/4+1250*xx*yy*dx^2;
    end
end
```

OUTPUT:

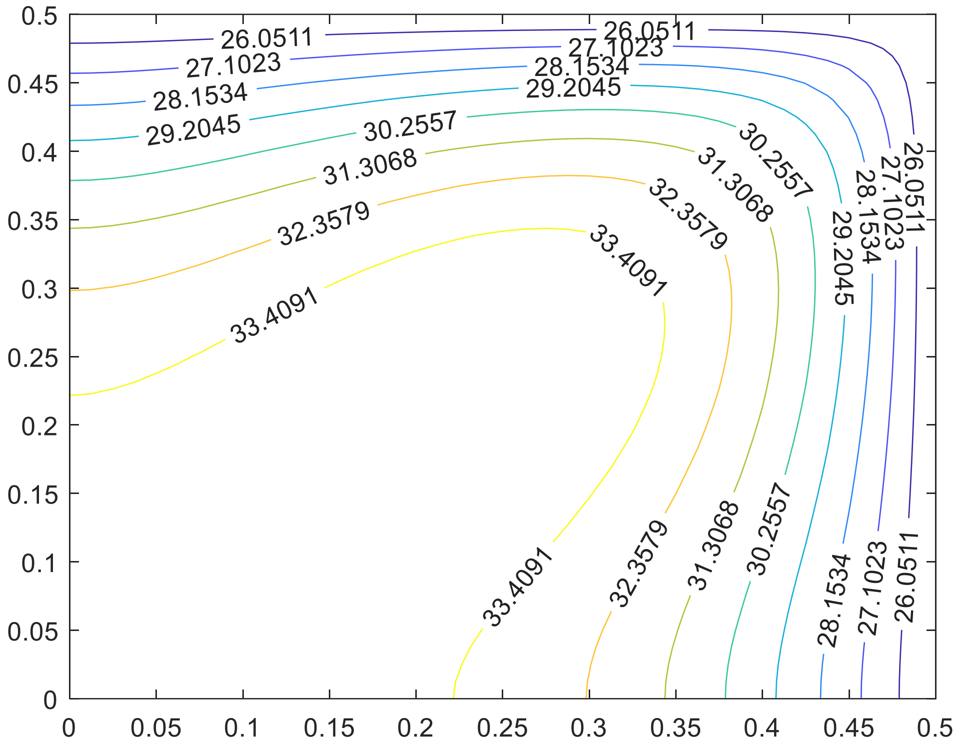




35
34
33
32
31
30
29
28
27
26
25
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
0.5
0



35
34
33
32
31
30
29
28
27
26
25
0.5
0
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0



26.0511
27.1023
28.1534
29.2045
30.2557
31.3068
32.3579
33.4091
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5