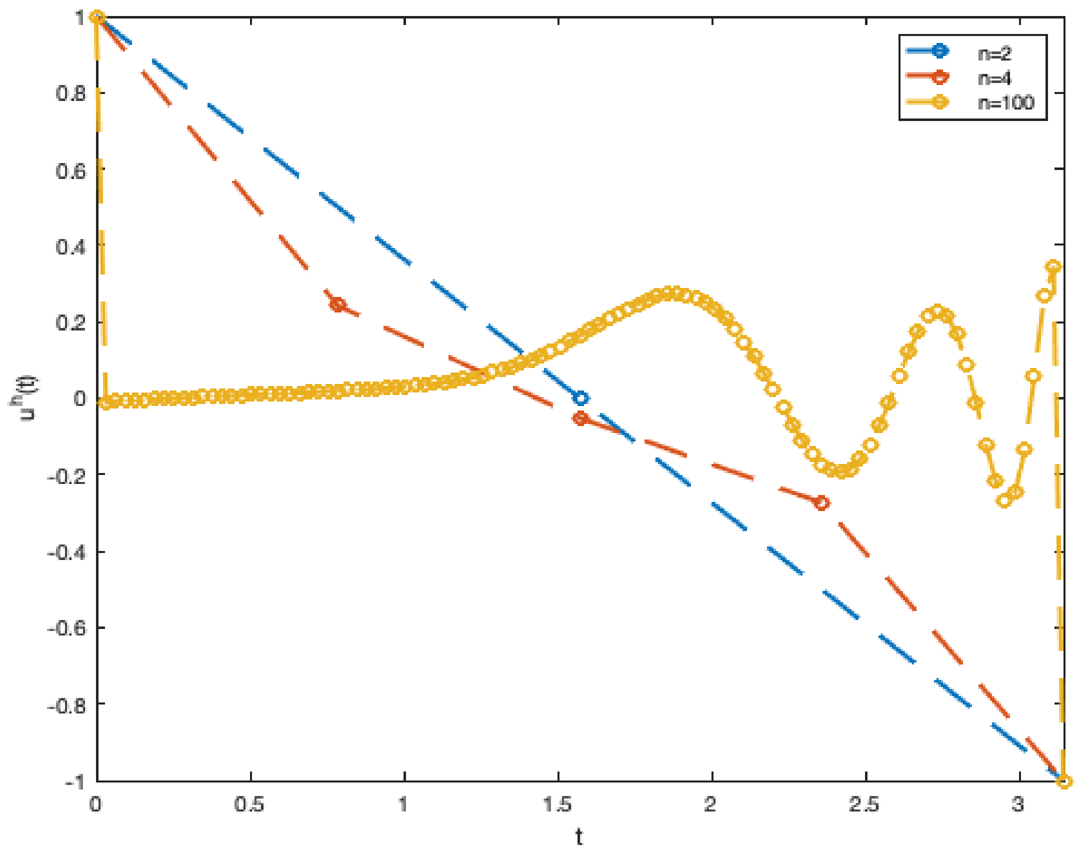
```
clc;clear all;close all


n=[2,4,100]; % n values being considered

for i=1:length(n)
    [U,t]=Nscheme(n(i));
    plot(t,U,'--o',LineWidth=2),hold on

end
legend('n=2','n=4','n=100')
xlabel('t')
ylabel('u^h(t)')
axis tight



function [U,t]=Nscheme(n)

h=pi/n; % step size


% Entries of matrix A
for i=2:n
    ti=(i-1)*h; % i=1,2,3,...n-1

    % diagonal entries
    A(i-1,i-1)=(1/h)*(sin(ti-h/2)+sin(ti+h/2))+2*h*sin(ti);

    % upper diagonal entries
    A(i-1,i)=-1/h*(sin(ti-h/2));

    % lower diagonal entries
    A(i,i-1)=A(i-1,i);
end

%Reshaping matrix A to by of size n-1 by n-1
A=A(1:n-1,1:n-1);

% creating the time vector
for i=1:n+1
    t(i)=(i-1)*h; % i = 0,1,2,....n
end

% Initializing the b vector

b=zeros(1,size(A,1))';
for i=2:n
    b(i-1)=2*h*sin(2*t(i))+(1/pi)*(sin(t(i-1))-sin(t(i+1)))+(4*h*t(i)/
pi-2*h)*sin(t(i));
end
```

```
c=A\b; % coefficients of the Galerkin approximation


% initializing the the numerical approximation storage space
U=zeros(1,n+1);

% Boundary Values
U(1)=1;U(end)=-1;


% Implementation of equation (2)
for k=2:length(t)-1

    for i=2:length(t)-1
        U(k)=U(k)+hat(i,t,t(k),h)*c(i-1);
    end

    U(k)=U(k)+(hat0(k,t,h)-hatn(k,t,h));

end

end


%%%% The hat functions %%%%%
function phi=hat(i,t,t_i,h)
    phi=0;
    % instances where phi is non-zero
    if (t(i-1)<=t_i) && (t_i<=t(i))
        phi=(t_i-t(i-1))/h;
    elseif (t(i)<=t_i) && (t_i<=t(i+1))
        phi=(t(i+1)-t_i)/h;
    end
end

function phin=hatn(i,t,h)
    phin=0;
     % instance where phin is non-zero
    if (t(end-1)<=t(i)) && (t(i)<=t(end))
        phin=(t(i)-t(end-1))/h;
    end
end


 function phi0=hat0(i,t,h)
    phi0=0;
    % instance where phi0 is non-zero
     if (t(1)<=t(i)) && (t(i)<=t(2))
         phi0=(t(2)-t(i))/h;
      end
 end
```



Published with MATLAB® R2022a