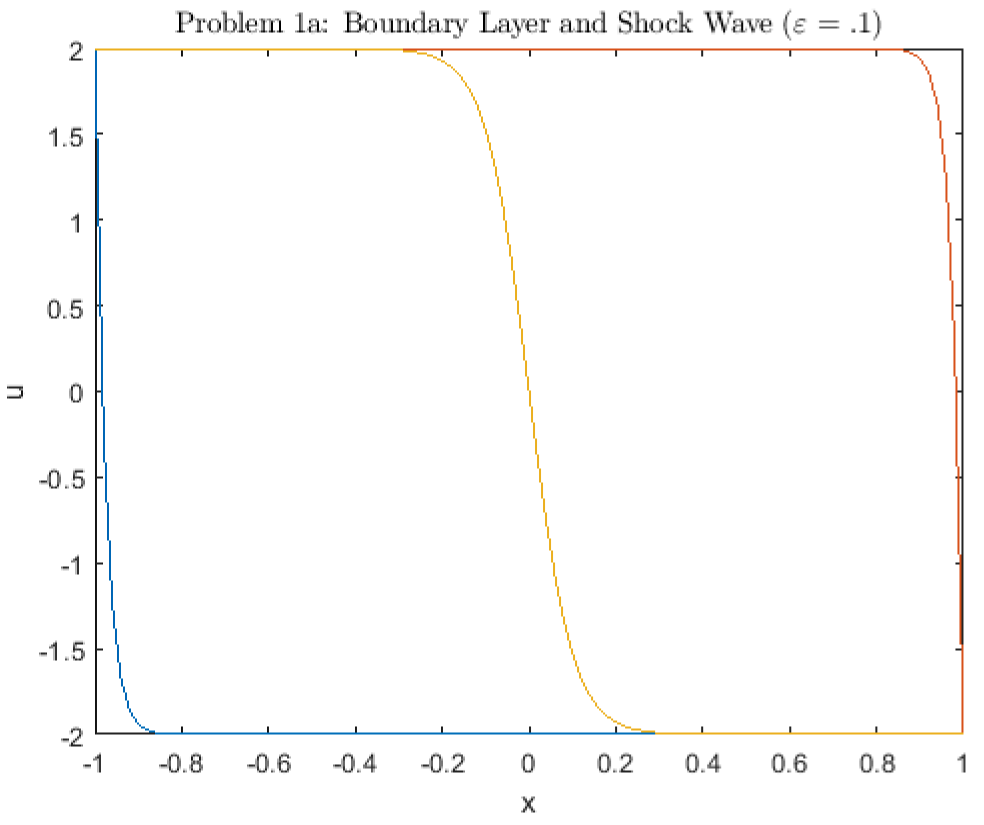
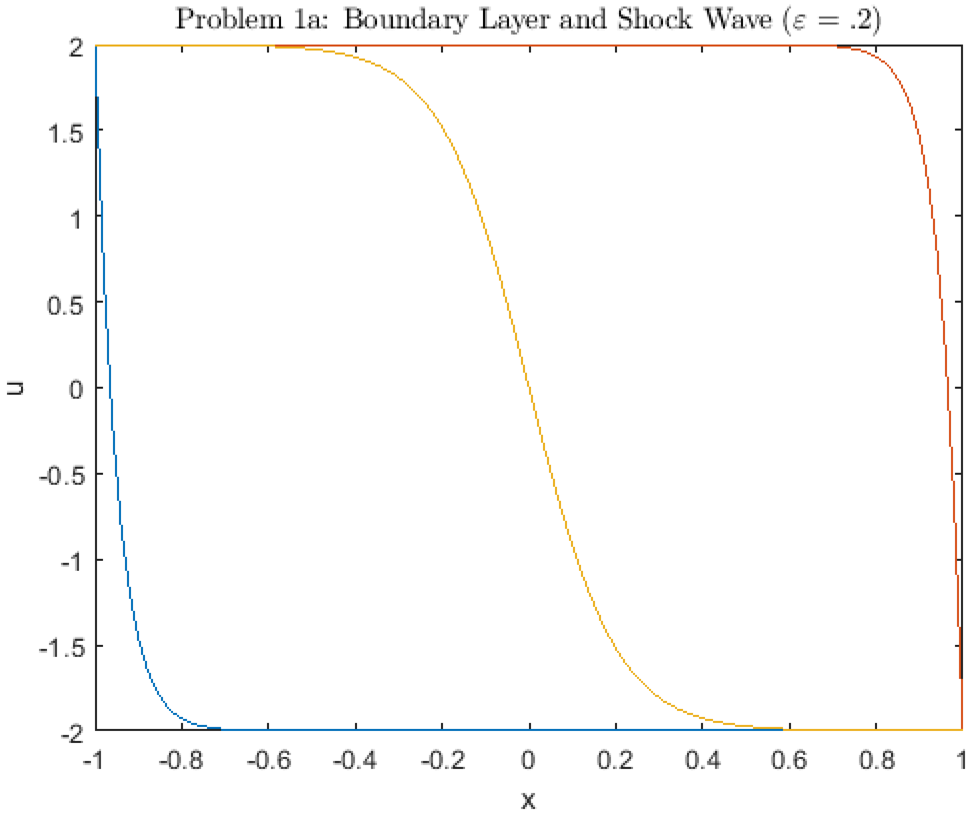
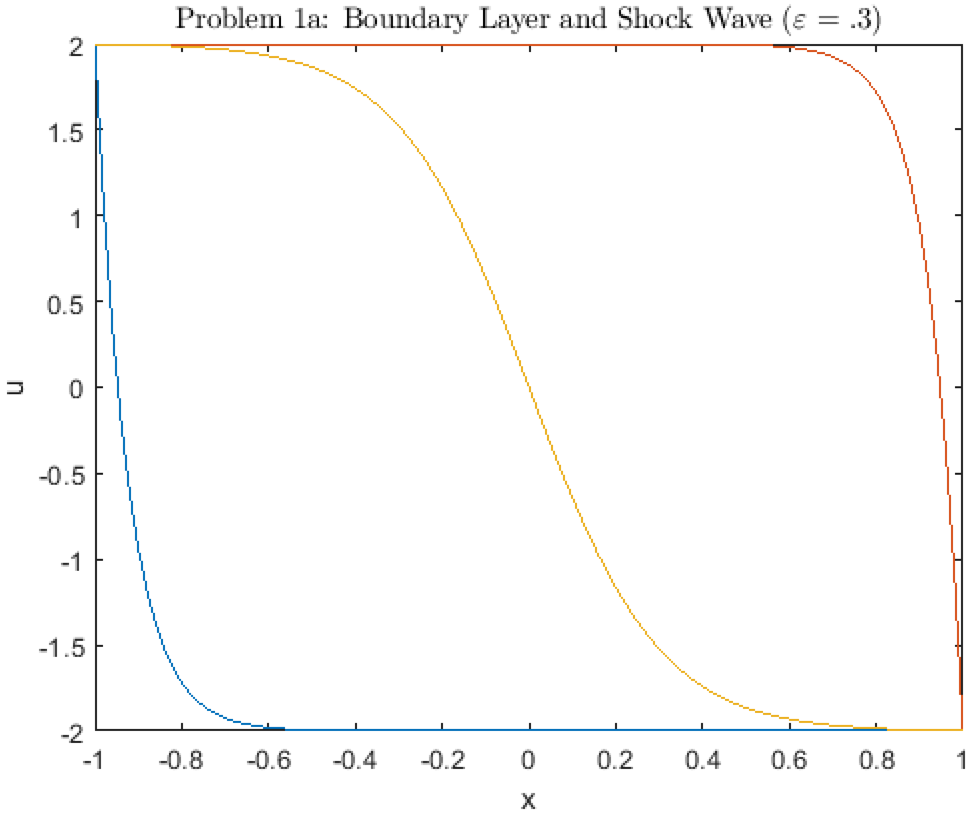
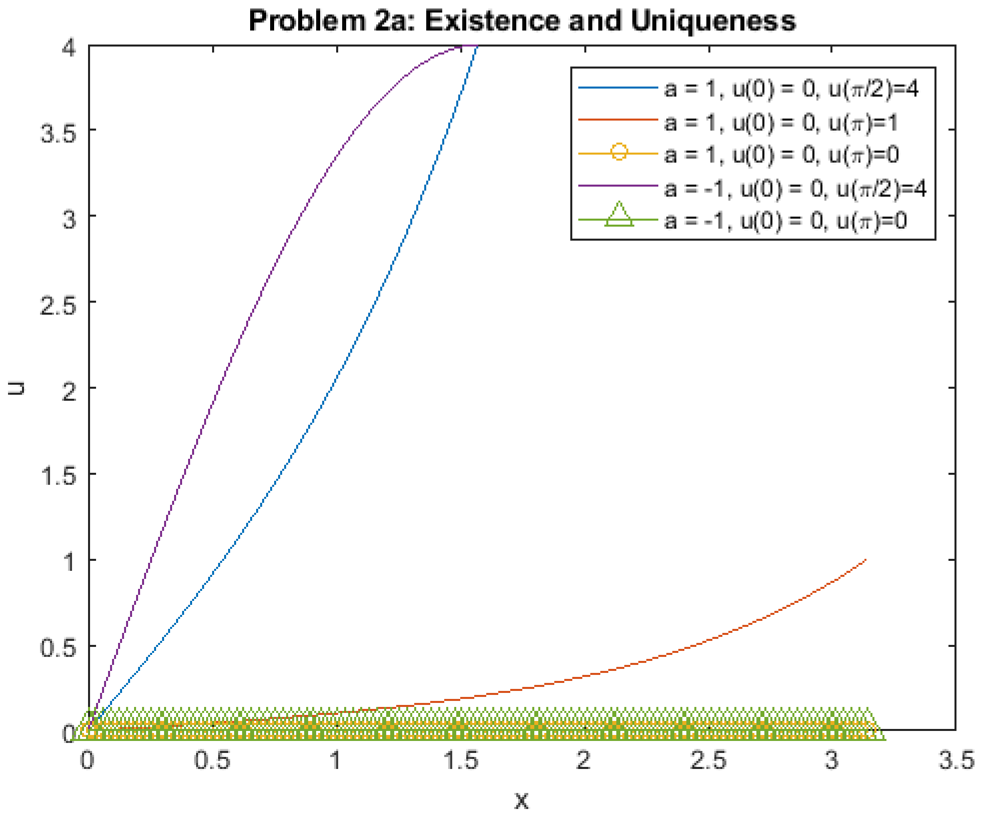
```
clc, clear;
close all;

n = 101;
%making the grid
x = linspace(-1,1,n);
delta = 2/n;

u1 = bvp(-4,.1,delta,n);
u2 = bvp(4,.1,delta,n);
u3 = bvpSpecial(0,.1,delta,n);
u4 = bvp(-4,.2,delta,n);
u5 = bvp(4,.2,delta,n);
u6 = bvpSpecial(0,.2,delta,n);
u7 = bvp(-4,.3,delta,n);
u8 = bvp(4,.3,delta,n);
u9 = bvpSpecial(0,.3,delta,n);

figure('Name',"epsilon .1")
plot(x,u1);
hold on;
plot(x,u2);
hold on;
plot(x,u3);
title('Problem 1a: Boundary Layer and Shock Wave ($\varepsilon$
 = .1)','Interpreter','latex')
xlabel('x')
ylabel('u')

figure('Name',"epsilon .2")
plot(x,u4);
hold on;
plot(x,u5);
hold on;
plot(x,u6);
title('Problem 1a: Boundary Layer and Shock Wave ($\varepsilon$
 = .2)','Interpreter','latex')
xlabel('x')
ylabel('u')

figure('Name',"epsilon .3")
plot(x,u7);
hold on;
plot(x,u8);
hold on;
plot(x,u9);
title('Problem 1a: Boundary Layer and Shock Wave ($\varepsilon$
 = .3)','Interpreter','latex')
xlabel('x')
ylabel('u')

[g1,u21] = uniqueness(.2,1,n,1);
```

```
[g2,u22] = uniqueness(.2,1,n,2);
[g3,u23] = uniqueness(.2,1,n,3);
[g4,u24] = uniqueness(.2,-1,n,1);
[g5,u25] = uniqueness(.2,-1,n,2);
[g6,u26] = uniqueness(.2,-1,n,3);
figure('Name','2A')
plot(g1,u21, g3,u23)
hold on;
plot(g2,u22,"Marker","o")
hold on;
plot(g4,u24)
hold on;
plot(g5,u25,"Marker","^",MarkerSize=12)
title('Problem 2a: Existence and Uniqueness')
legend('a = 1, u(0) = 0, u(\pi/2)=4','a = 1, u(0) = 0, u(\pi)=1',...
     'a = 1, u(0) = 0, u(\pi)=0','a = -1, u(0) = 0, u(\pi/2)=4',...
     'a = -1, u(0) = 0, u(\pi)=0')
xlabel('x')
ylabel('u')

figure('Name', 'failed unique')
plot(g6,u26)
title('Problem 2a: Invalid Case')
legend('a = -1, u(0) = 0, u(\pi)=1')
xlabel('x')
ylabel('u')

function u = bvp(f,eps,delta,n)
    % Calculating derived coefficients
    iMinusOneCoeff = -(f/(2*delta)+eps/delta^2);
    iCoeff = 2*eps/delta^2;
    iPlusOneCoeff = f/(2*delta)-eps/delta^2;

    %Constructing arrays for tridiagonal matrix
    a = iMinusOneCoeff.*ones(n,1); b = iCoeff.*ones(n,1); c =
 iPlusOneCoeff.*ones(n,1);
    b(1) = 1; c(1) = 0; a(n) = 0; b(n) = 1;
    d = zeros(n,1); d(1)=2; d(n) = -2;

    %Solving tridiagonal system
    u = THOMAS3(a,b,c,d,n);
end

% the f parameter is the guess value
function u = bvpSpecial(f,eps,delta,n)
    tol = 1e-7;
    correction = 1;
    unew = f.*ones(1,n);

    %Iterate through the guess until met tolerance threshold
    while correction > tol
        %Constructing arrays for tridiagonal matrix
        a = -(f/(2*delta)+eps/delta^2).*ones(1,n);
        b = (2*eps/delta^2).*ones(1,n);
        c = f/(2*delta)-eps/delta^2.*ones(1,n);

        b(1) = 1; c(1) = 0; a(n) = 0; b(n) = 1;
        d = zeros(n,1); d(1)=2; d(n) = -2;

        %Solving tridiagonal system
        unew = THOMAS3(a,b,c,d,n);
        correction = max(abs(unew-f));
        f = unew;
    end
    u = unew;
end

%f parameter is guess value
%coe parameter is given 'a' in the problem statement
% option from 1 to 3 to set different given boundary values
function [g,u] = uniqueness(f, coe, n, option)

    switch option
        case 1
            % Dicrestize the grid
            g = linspace(0,pi/2,101);
            delta = (pi/2)/n;
            tol = 1e-7;
            correction = 1;
            unew = f.*ones(1,n);

            %Iterate through the guess until met tolerance threshold
            while correction > tol
                %Constructing arrays for tridiagonal matrix
                a = (1/delta^2)./f.*ones(1,n);
                b = (-2/delta^2)./f.*ones(1,n);
                c = (1/delta^2)./f.*ones(1,n);

                b(1) = 1; c(1) = 0; a(n) = 0; b(n) = 1;
                d = coe.*ones(n,1); d(1)=0; d(n) = 4;

                %Solving tridiagonal system
                unew = THOMAS3(a,b,c,d,n);
                correction = max(abs(unew-f));
                f = unew;
            end
            u = unew;
        case 2
            % Dicrestize the grid
            g = linspace(0,pi,101);
            delta = (pi)/n;
            tol = 1e-7;
            correction = 1;
            unew = f.*ones(1,n);

            %Iterate through the guess until met tolerance threshold
            while correction > tol
                %Constructing arrays for tridiagonal matrix
                a = (1/delta^2)./f.*ones(1,n);
                b = (-2/delta^2)./f.*ones(1,n);
                c = (1/delta^2)./f.*ones(1,n);

                b(1) = 1; c(1) = 0; a(n) = 0; b(n) = 1;
                d = coe.*ones(n,1); d(1)=0; d(n) = 0;

                %Solving tridiagonal system
                unew = THOMAS3(a,b,c,d,n);
                correction = max(abs(unew-f));
                f = unew;
            end
            u = unew;
        case 3
            % Dicrestize the grid
            g = linspace(0,pi,101);
            delta = (pi)/n;
            tol = 1e-7;
            correction = 1;
            unew = f.*ones(1,n);

            %Iterate through the guess until met tolerance threshold
            while correction > tol

                %Constructing arrays for tridiagonal matrix
                a = (1/delta^2)./f.*ones(1,n);
                b = (-2/delta^2)./f.*ones(1,n);
                c = (1/delta^2)./f.*ones(1,n);

                b(1) = 1; c(1) = 0; a(n) = 0; b(n) = 1;
                d = coe.*ones(n,1); d(1)=0; d(n) = 1;

                %Solving tridiagonal system
                unew = THOMAS3(a,b,c,d,n);
                correction = max(abs(unew-f));
                f = unew;
            end
            u = unew;
    end

end
% supporting function
function x = THOMAS3(a,b,c,d,n)

    %initial condition
    bbar(1) = b(1);
    cbar(1) = c(1);
    dbar(1) = d(1);

    %making upper triangle
    for i = 2:n
        multiplier = a(i)./bbar(i-1);
        abar(i) = a(i) - bbar(i-1).*multiplier;
        bbar(i) = b(i) - cbar(i-1).*multiplier;
        cbar(i) = c(i);
        dbar(i) = d(i) - dbar(i-1).*multiplier;
    end

    %initialize x of size n
    x = ones(1,n);

    %initialize end condition
    x(n) = dbar(n)/bbar(n);

    % Upward substitution AKA zip it up
    for i = n-1:-1:1
        x(i) = (dbar(i)-(cbar(i)*x(i+1)))/bbar(i);
    end
end
```

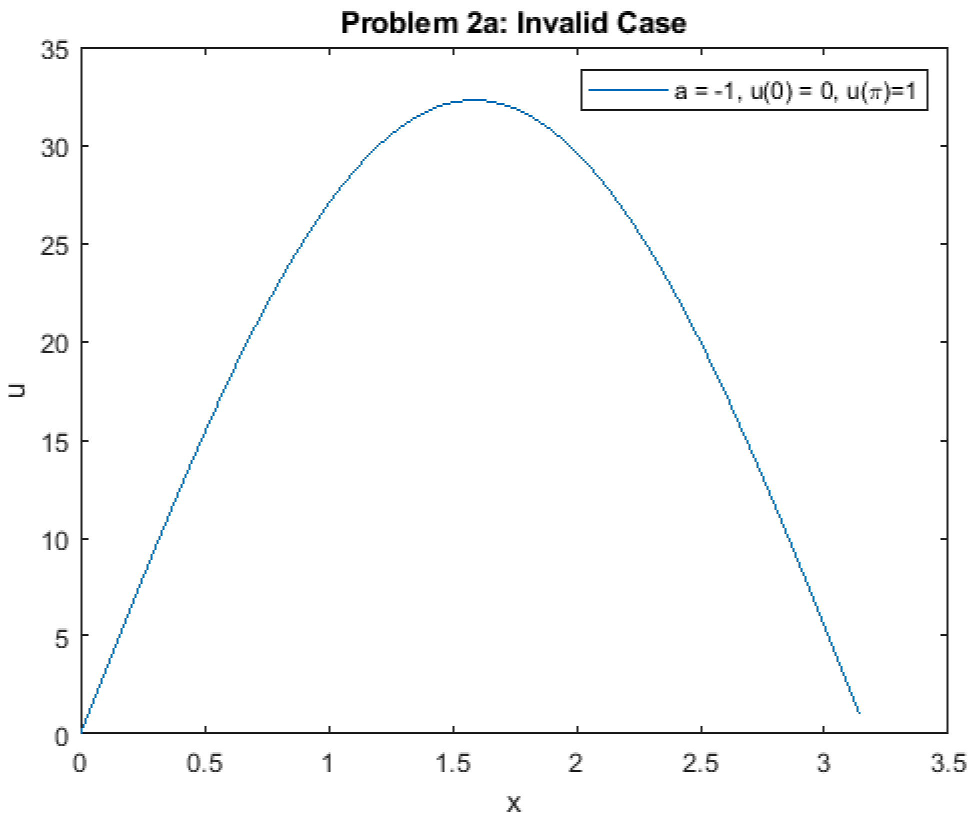



Problem 1a: Boundary Layer and Shock Wave ($\varepsilon = .2$)
u
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Problem 1a: Boundary Layer and Shock Wave ($\varepsilon = .3$)
u
x
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1



Problem 2a: Existence and Uniqueness
a = 1, u(0) = 0, u(π/2)=4
a = 1, u(0) = 0, u(π)=1
a = 1, u(0) = 0, u(π)=0
a = -1, u(0) = 0, u(π/2)=4
a = -1, u(0) = 0, u(π)=0
u
x



Problem 2a: Invalid Case
a = -1, u(0) = 0, u(π)=1
u
x

Published with MATLAB® R2022b