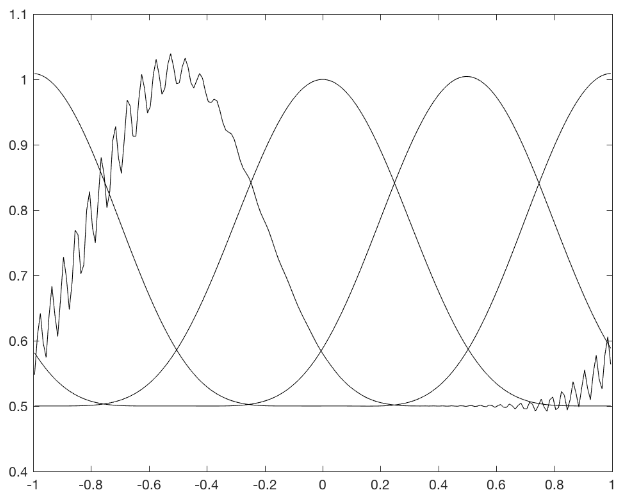
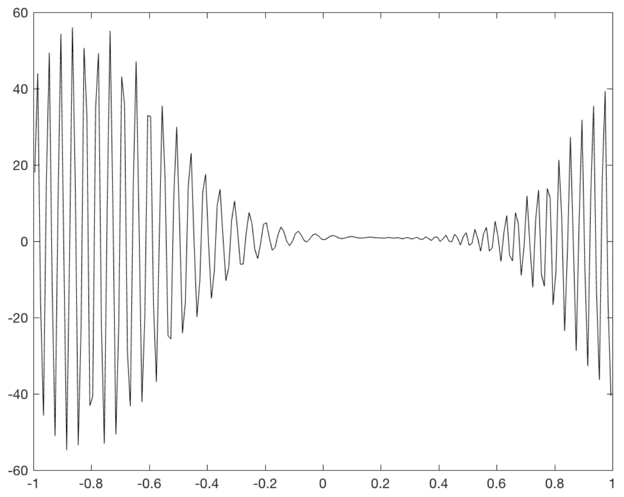
Caleb Logemann
AER E 546 Fluid Mechanics and Heat Transfer I
Homework 8

1. The following is my code for the linear combination of upwinding and downwinding.

```
function [u] = upwindDownwind(u0, f, p, deltaX, deltaT, nTimeSteps)

    nCells = length(u0);
    u = zeros(nTimeSteps+1, nCells);
    u(1,:) = u0;

    % vector to store fluxes at cell interfaces
    % F(j) is flux at cell interface x_{j-1/2} on left hand side
    F = zeros(nCells + 1);
    nu = deltaT/deltaX;

    for n = 1:nTimeSteps
        % Compute fluxes
        for j = 1:nCells
            jm1 = j-1;
            % periodic boundary conditions
            if (j == 1)
                jm1 = nCells;
            end
            F(j) = p*f(u(n, jm1)) + (1 - p)*f(u(n, j));
        end

        for j = 1:nCells
            jp1 = j+1;
            if(j == nCells)
                jp1 = 1;
            end
            u(n+1, j) = u(n, j) - nu*(F(jp1) - F(j));
        end
    end
end
```

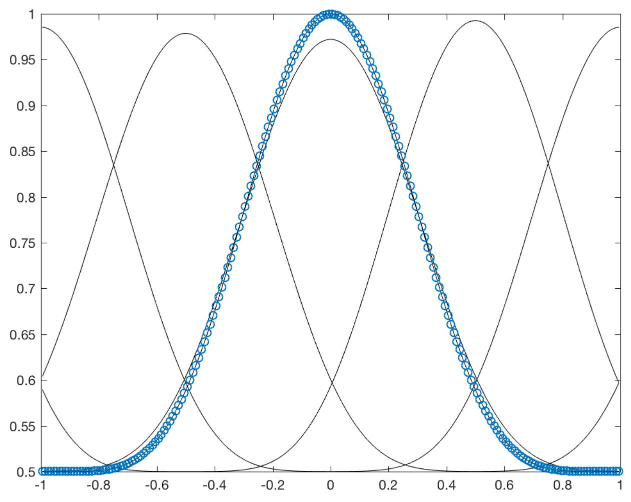
#1a The plot for $p = 0.5$ and $c = 0.3$ is shown below. Initially the solution is represented well, but eventually the solution diverges.





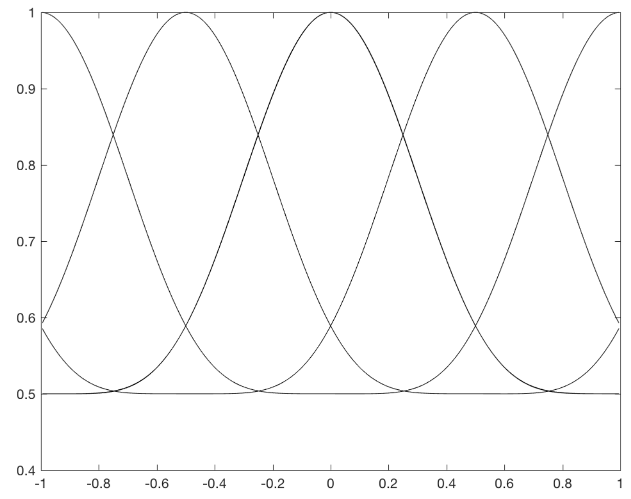
#1b The solution diverges because with $p = 0.5$ this is identical to the central differencing scheme, which we know to be unstable for all cfl conditions.

#2a For $p = 1.0$ we have complete upwinding and the solution plots are



#2b The simulated solution and the exact solution at $t = 2.0$ differ because upwinding is diffusive, so the numerical solution has decayed/diffused some instead of just convecting.

#3a For $p = 0.55$ and $c = 0.1$ the following plot is produced.



#3b I found that the solution was accurate at $t = 2.0$ up to $c = 0.33$.

#4

2. The following two functions evaluate the RHS for the central and upwinding discretizations. These functions are used with the RK2 function that we have been using.

```
function [rhs] = central(u, f, deltaX)
    nCells = length(u);
    rhs = zeros(1,nCells);

    % vector to store fluxes at cell interfaces
    % F(j) is flux at cell interface x_{j-1/2} on left hand side
    F = zeros(nCells + 1);
    nu = 1/deltaX;
    % Compute fluxes
    for j = 1:nCells
        jm1 = j-1;
        % periodic boundary conditions
        if (j == 1)
            jm1 = nCells;
        end
        F(j) = 0.5*(f(u(jm1)) + f(u(j)));
    end

    for j = 1:nCells
        jp1 = j+1;
        if(j == nCells)
            jp1 = 1;
        end
        rhs(j) = nu*(F(j) - F(jp1));
    end
end
```

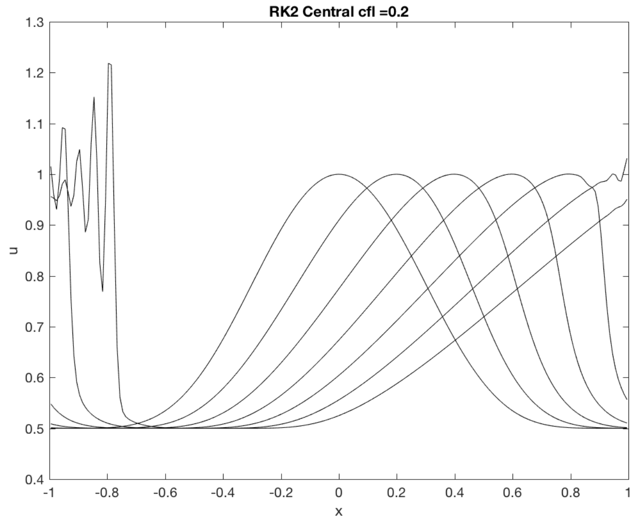
```
function [rhs] = upwind(u, f, deltaX)
    nCells = length(u);
    rhs = zeros(1,nCells);

    % vector to store fluxes at cell interfaces
    % F(j) is flux at cell interface x_{j-1/2} on left hand side
    F = zeros(nCells + 1);
    nu = 1/deltaX;

    % Compute fluxes
    for j = 1:nCells
        jm1 = j-1;
        % periodic boundary conditions
        if (j == 1)
            jm1 = nCells;
        end
        F(j) = f(u(jm1));
    end

    for j = 1:nCells
        jp1 = j+1;
        if(j == nCells)
            jp1 = 1;
        end
        rhs(j) = nu*(F(j) - F(jp1));
    end
end
```

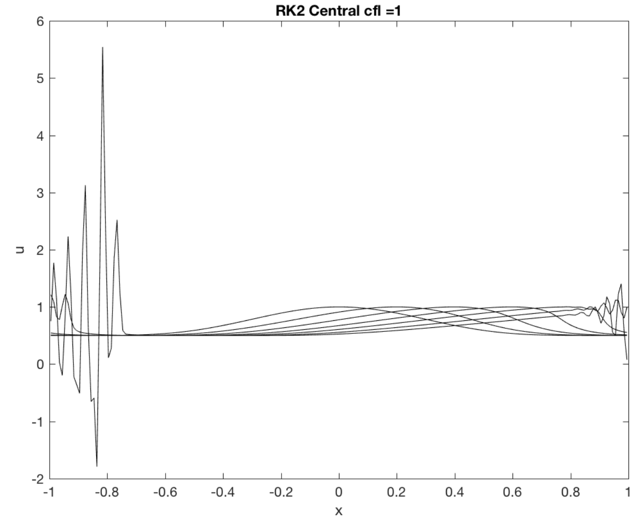
These next two function run the Lax-Wendroff and MacCormack methods.

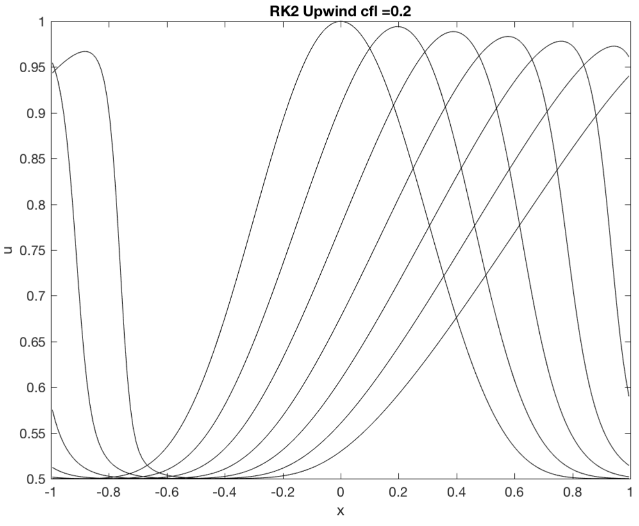
```
function [u] = laxWendroff(f, df, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;
    a = 0.5 * deltaT/deltaX;
    b = 0.5 * (deltaT/deltaX)^2;

    for n = 1:nTimeSteps
        for j = 1:nGridCells
            % periodic boundary conditions
            jm1 = j-1;
            if (j == 1)
                jm1 = nGridCells;
            end
            jp1 = j+1;
            if (j == nGridCells)
                jp1 = 1;
            end

            % update
            % traditional laxWendroff
            Ap = df(0.5*(u(n, j) + u(n, jp1)));
            Am = df(0.5*(u(n, j) + u(n, jm1)));
            fc = f(u(n,jp1)) - f(u(n,jm1));
            fl = f(u(n,j)) - f(u(n,jm1));
            fr = f(u(n,jp1)) - f(u(n,j));
            u(n+1, j) = u(n, j) - a*fc + b*(Ap*fr - Am*fl);
        end
    end
end
```

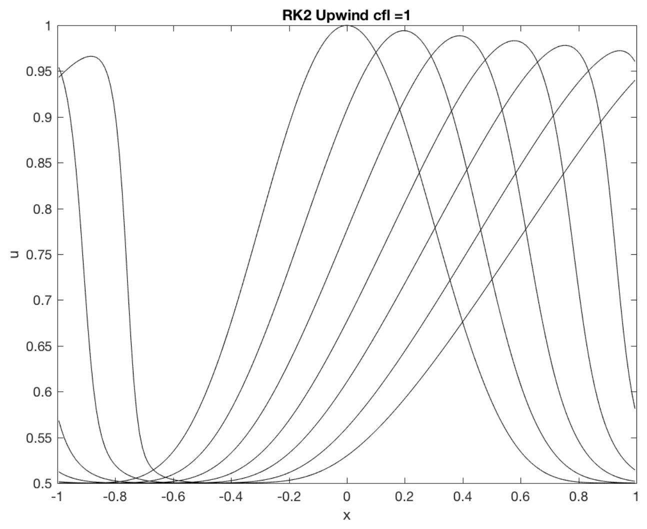
```
function [u] = maccormack(f, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;

    a = deltaT/deltaX;
    b = 0.5 * deltaT/deltaX;

    F = zeros(nGridCells,1);
    Ustar = zeros(nGridCells,1);

    for n = 1:nTimeSteps
        % Fluxes
        for j = 1:nGridCells
            F(j) = f(u(n, j));
        end

        % Ustar
        for j = 1:nGridCells-1
            Ustar(j) = u(n, j) - a*(F(j+1) - F(j));
        end
        % periodic boundary conditions
        Ustar(nGridCells) = u(n, 1);

        % Ustar Fluxes
        for j = 1:nGridCells-1;
            F(j) = f(Ustar(j));
        end

        % update solution
        % periodic boundary condition
        u(n+1, 1) = 0.5*(u(n, nGridCells) + Ustar(nGridCells));
        for j = 2:nGridCells
            u(n+1,j) = 0.5*(u(n, j) + Ustar(j)) - b*(F(j) - F(j-1));
        end
    end
end
```

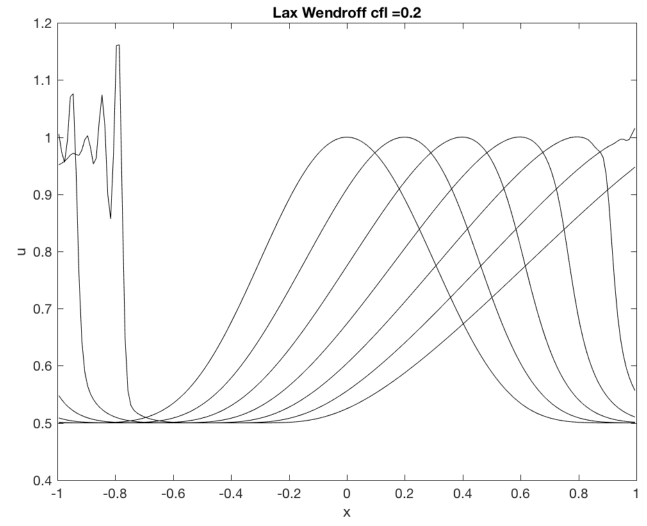
The following images are produced. Note that the central differencing is unstable, but the lower cfl improves this. The upwind method is stable and doesn't produce any oscillations. However the second order methods, MacCormack and Lax-Wendroff do produce oscillations after the shock occurs.

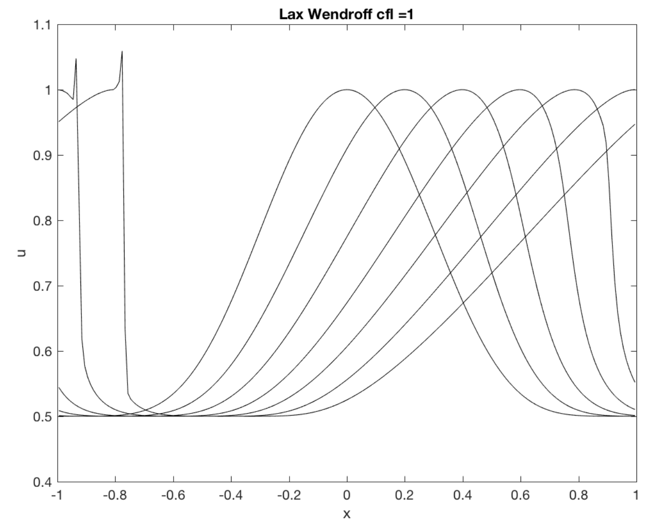


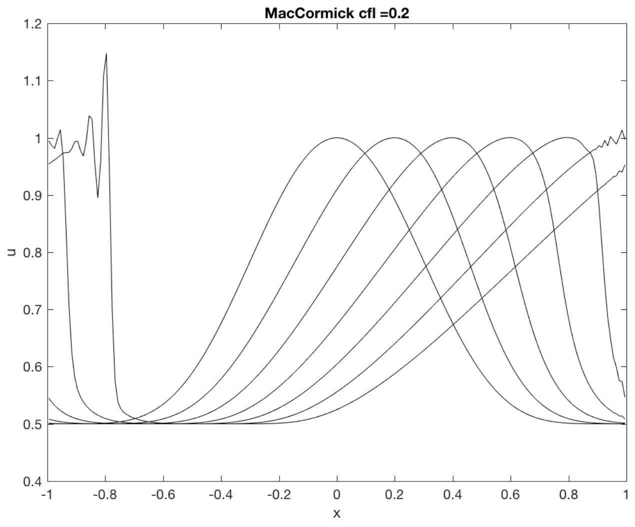


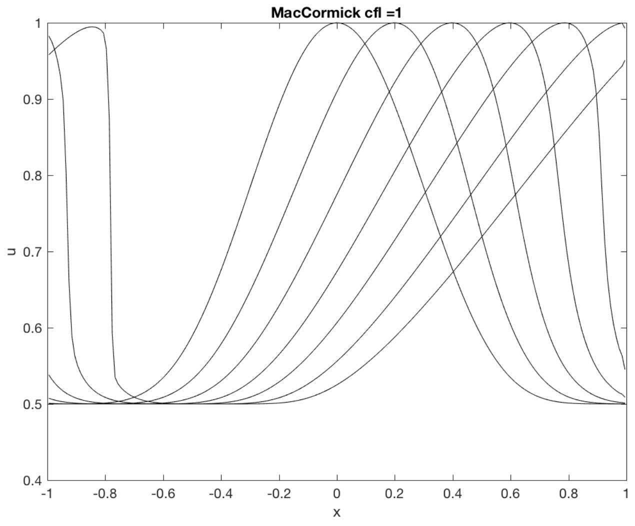



RK2 Upwind cfl =1
u
x


Lax Wendroff cfl =0.2
u
x


Lax Wendroff cfl =1
u
x



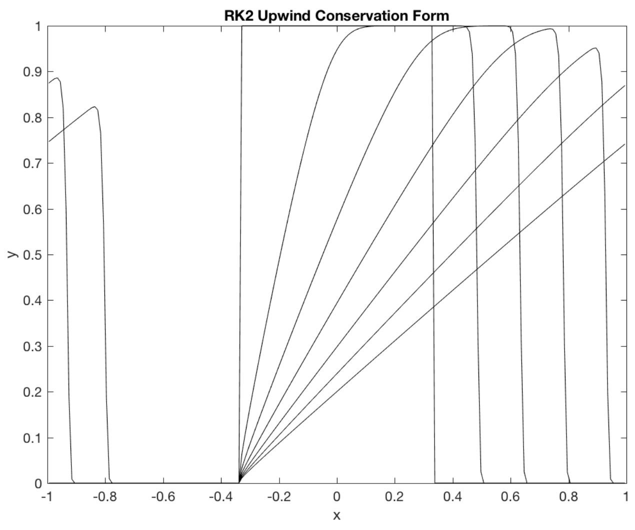


3. I used the same RK2 upwind in conservation form and Lax-Wendroff method as shown in problem 2. However for RK2 in convection form the following is my RHS function.

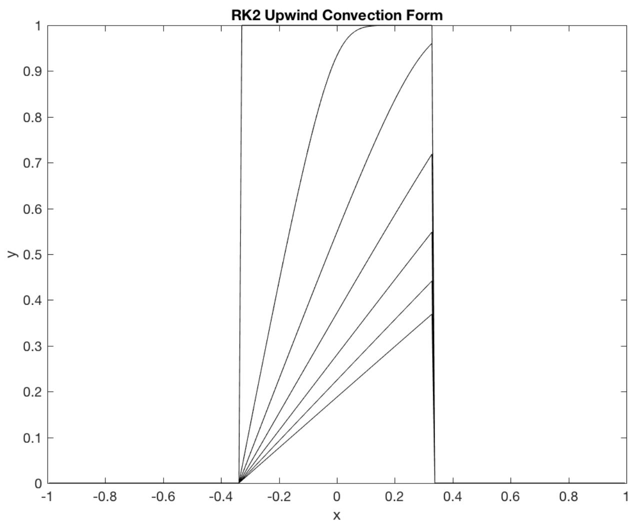
```
function [rhs] = upwindConvection(u, deltaX)
    nCells = length(u);
    rhs = zeros(1,nCells);

    nu = 1/deltaX;

    for j = 1:nCells
        jm1 = j-1;
        % periodic boundary conditions
        if (j == 1)
            jm1 = nCells;
        end
        rhs(j) = -u(j)*nu*(u(j) - u(jm1));
    end
end
```

For Burger's equation the shock speed is the average of the upper and lower states, so for this problem the shock velocity will be 1/2.

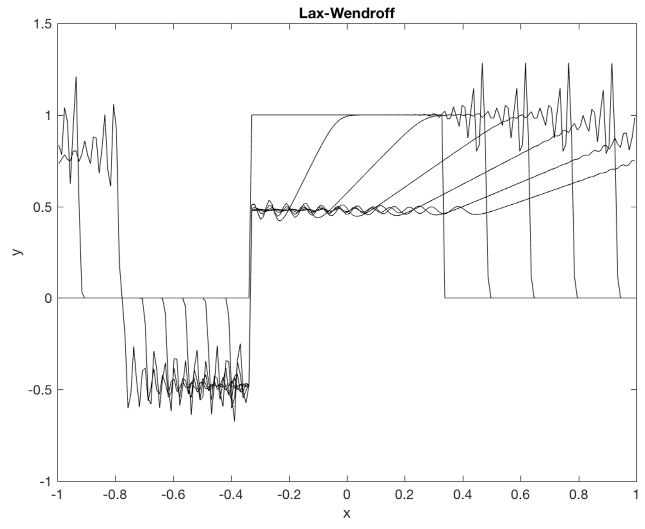
The following images are produced. The RK2 method with upwinding in conservation form is stable and the shock moves at the correct speed. The diffusion of the full height of the wave can be seen.



For the RK2 method with upwinding in convection form the shock does not travel at the correct velocity. In fact the shock doesn't travel at all. This method is stable and doesn't blow up, but it doesn't converge to the correct solution.



The Lax-Wendroff method propogates the shock at the correct speed, but it does not properly find the rarefaction/expansion. Also the Lax-Wendroff has spurious oscillations around the shocks.


Lax-Wendroff
1.5
1
0.5
0
-0.5
-1
y
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
x