Caleb Logemann
MATH667 Hyperbolic Partial Differential Equations
Homework 2

Solve Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x = 0 \qquad x \in [-5, 5] \tag{1}$$

$$u(x, 0) = \begin{cases} u_l & x \leq 0 \\ u_r & x > 0 \end{cases} \tag{2}$$

with the following Riemann initial data sets.

(a)

$$u(x, 0) = \begin{cases} 1 & x \leq 0 \\ -0.5 & x > 0 \end{cases}$$

(b)

$$u(x, 0) = \begin{cases} -1 & x \leq 0 \\ 0.5 & x > 0 \end{cases}$$

1. Determine the exact solutions for all $t > 0$.

   (a) In this case $u_l > u_r$, so the solution is a shock that propogates with speed

   $$s = \frac{f(u_l) - f(u_r)}{u_l - u_r} = \frac{1}{2}(u_l + u_r) = \frac{1}{2}(1 - 0.5) = 0.25$$

   Therefore the exact solution is

   $$u(x, t) = \begin{cases} 1 & x \leq 0.25t \\ -0.5 & x > 0.25t \end{cases}$$

   (b) In this case $u_l < u_r$ so the solution is a rarefaction. Therefore the left side will move with speed u_l, the right side will move with speed u_r, and the middle with connect the sides linearly.

   $$u(x, t) = \begin{cases} -1 & x \leq -t \\ x/t & -t < x < 0.5t \\ 0.5 & 0.5t \leq x \end{cases}$$

2. The following function implement the Roe scheme, the Lax-Friedrichs scheme, and both the standard and Richtmyer versions of the Lax-Wendroff method. I implemented both the standard Lax-Wendroff and the Richtmyer methods because the Lax-Wendroff method was giving me the wrong weak solution, while the Richtmyer was giving me the entropy solution in the rarefaction case.

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

    % flux array, F(i) is flux at i - 1/2 interface
```

```
    % periodic boundary conditions F(nGridCells + 1) = F(1), so F(nGridCells + 1) not
        ↪ necessary
    F = zeros(nGridCells,1);

    for n = 1:nTimeSteps
        % compute fluxes at boundaries
        for j = 1:nGridCells
            % zero flux boundary conditions
            jm1 = j-1;
            if (j == 1)
                jm1 = 1;
            end

            ful = f(u(n,jm1));
            fur = f(u(n, j));
            if ((ful - fur)/(u(n,jm1) - u(n,j)) >= 0)
                F(j) = ful;
            else
                F(j) = fur;
            end
        end

        % update solution
        for j = 1:nGridCells
            % zero flux boundaryH03 conditions
            jp1 = j+1;
            if (j == nGridCells)
                jp1 = nGridCells;
            end

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

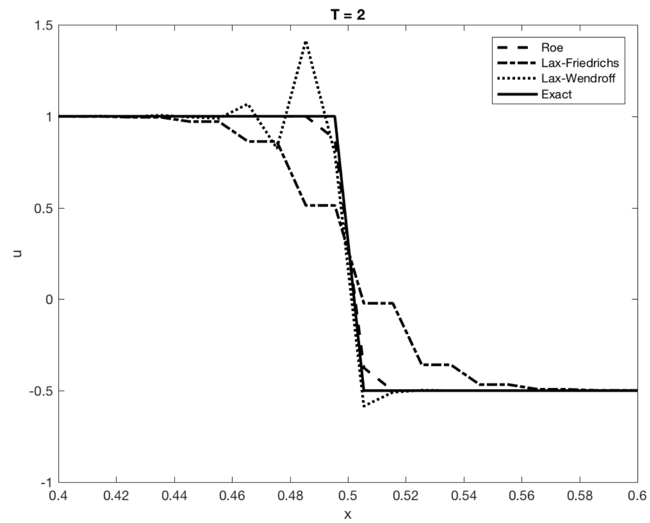
```
function [u] = laxFriedrichs(f, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;
    a = 0.5 * deltaT/deltaX;

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

            % update
            % u(n+1, j) = u(n, j) - a*(u(n,jp1) - u(n,jm1)) + 0.5*(u(n,jp1) - 2*u(n,j)
                ↪ + u(n,jm1));
            u(n+1, j) = 0.5*(u(n,jm1) + u(n,jp1)) - a*(f(u(n,jp1)) - f(u(n,jm1)));
        end
    end
end
```

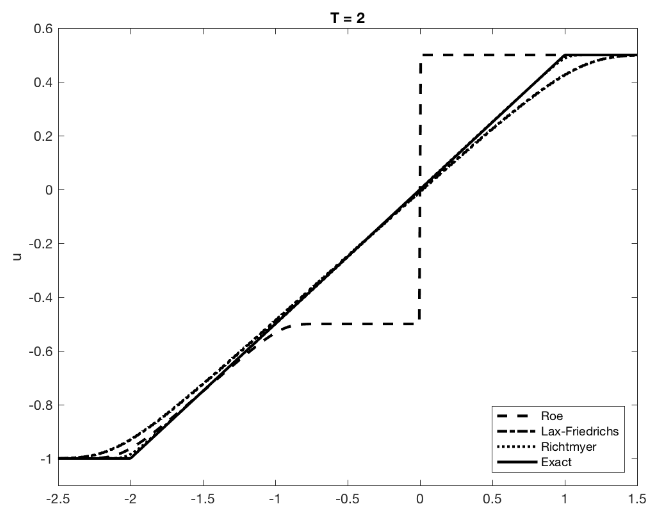
```
function [u] = laxWendroff(f, df, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;
    a = 0.5 * deltaT/deltaX;
    b = 0.5 * (deltaT/deltaX)^2;

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

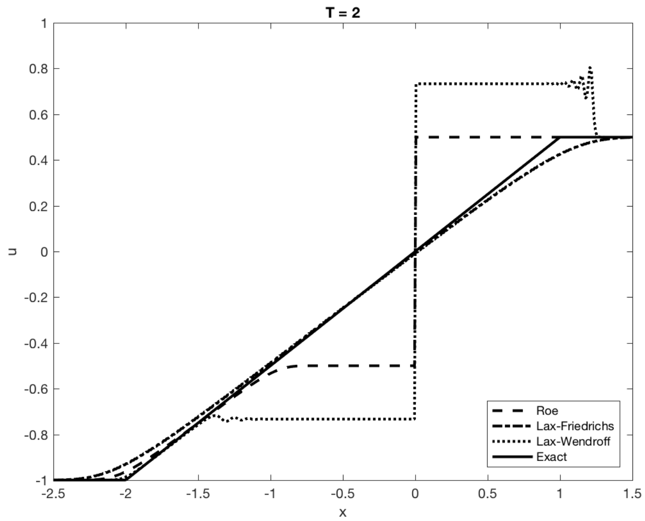
```
function [u] = richtmyer(f, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;

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

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

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

        % Uhalf
        % Uhalf(j) = U^(n+1/2)_(j+1/2)
        for j = 1:nGridCells-1
            Uhalf(j) = 0.5*(u(n, j)+u(n,j+1)) - b*(F(j+1) - F(j));
        end
        % zero flux boundary conditions
        Uhalf(nGridCells) = u(n, nGridCells);
```

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

        % update solution
        % zero flux boundary condition
        u(n+1, 1) = u(n, 1);
        for j = 2:nGridCells
            u(n+1,j) = u(n, j) - a*(F(j) - F(j-1));
        end
    end
end
```

The following script uses these functions to solve the Burgers equation for the shock and rarefaction case. The following images are produced. All of these images are zoomed in to the area of interest. Note that in rarefaction case the Richtmyer Lax-Wendroff method produces just a rarefaction were the standard Lax-Wendroff produces a shock and rarefaction. Also the oscillations in the Lax-Wendroff method cause this shock to be out of proportion.






T = 2
u
x
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
Roe
Lax-Friedrichs
Lax-Wendroff
Exact