Caleb Logemann
MATH667 Hyperbolic Partial Differential Equations
Homework 4

Solve Burgers' equation

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

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

with the following Riemann initial data sets.

(a)

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

(b)

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

1. Godunov's method uses an exact Riemann solver and is implemented in the following function.

```
function [u] = godunov(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
    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 (u(n, jm1) <= u(n, j))
                % min[ul < u < ur]{f(u)}
                % specific to burger's equation
                if (u(n, jm1)*u(n, j) < 0)
                    F(j) = 0;
                else
                    F(j) = min(ful, fur);
                end
            else
                % max[ur < u < ul]{f(u)}
                F(j) = max(ful, fur);
            end
        end
        % update solution
        for j = 1:nGridCells
            % zero flux boundary 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
```

The Local Lax-Friedrichs method uses the local maximum wave speed in the flux. Note that since we are taking the maximum of a linear function, I just took the maximum at the endpoints.

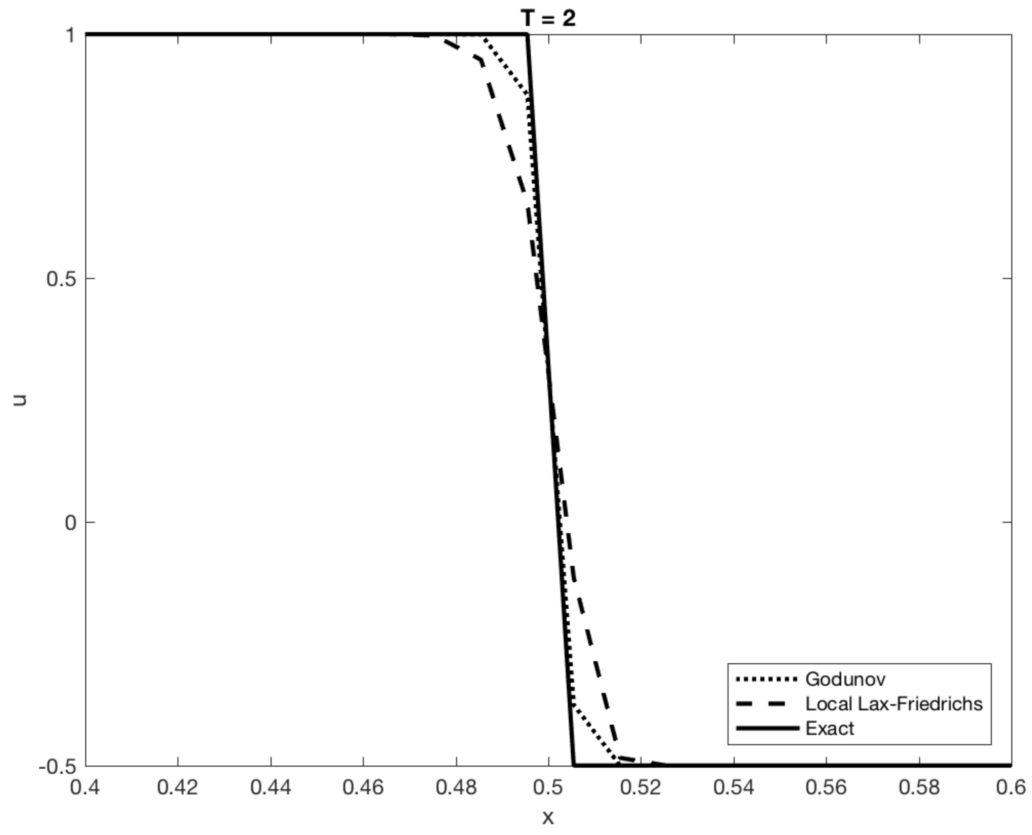
```
function [u] = localLaxFriedrichs(f, df, u0, deltaT, deltaX, nTimeSteps)
    nGridCells = length(u0);
    u = zeros(nTimeSteps+1, nGridCells);
    u(1, :) = u0;
    a = deltaT/deltaX;

    F = zeros(nGridCells);

    for n = 1:nTimeSteps
        % F(j) = f_{j-1/2}
        for j = 1:nGridCells
            % zero flux boundary conditions
            jm1 = j-1;
            if (j == 1)
                jm1 = 1;
            end
            alpha = max(abs(df(u(n, jm1))), abs(df(u(n, j))));
            F(j) = 0.5*(f(u(n,jm1)) + f(u(n, j)) - alpha*(u(n, j) - u(n, jm1)));
        end

        for j = 1:nGridCells
            % zero flux boundary conditions
            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) = u(n, j) - a*(F(jp1) - F(j));
        end
    end
end
```

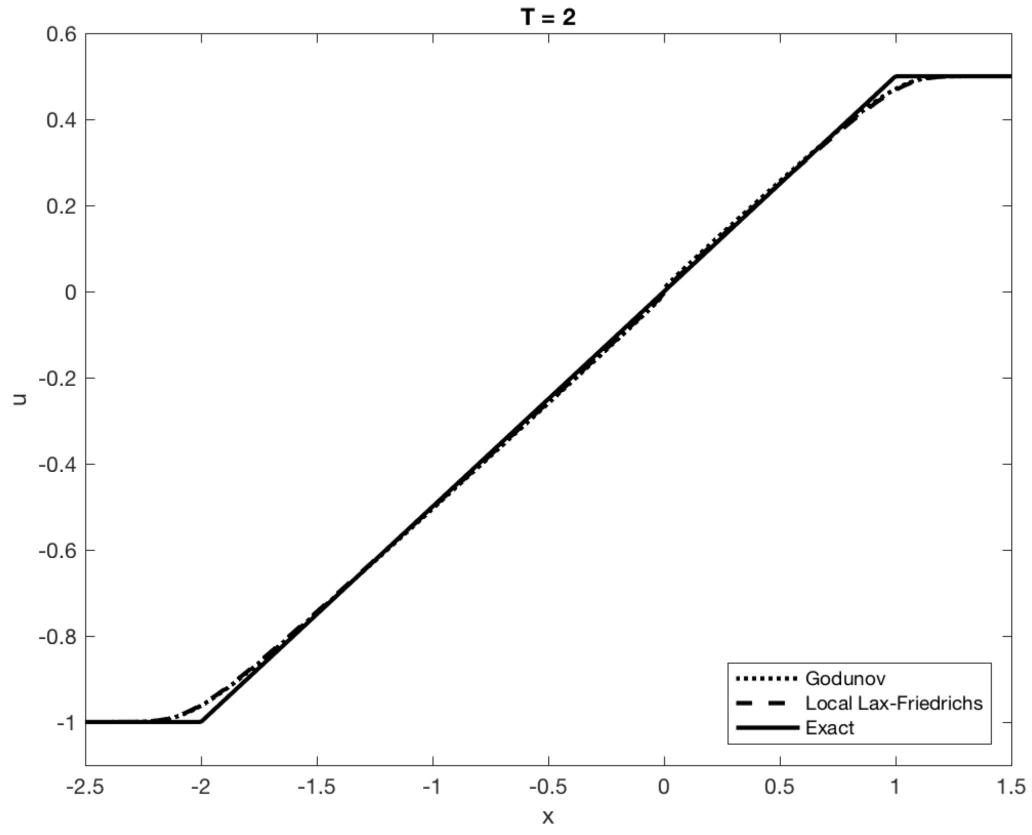
The following script now uses the previous two functions to solve the Riemann problems given previously.

```
%% 2 (a)
f = @(u) (0.5*u^2);
df = @(u) u;
u0func = @(x) 1.0*(x <= 0.0) - 0.5*(x > 0.0);
```

```
%u0func = @(x) -1.0*(x < 0.0) + 0.5*(x > 0.0);
deltaX = 0.01;
a = -5.0;
b = 5.0;
nGridCells = (b - a)/deltaX;

deltaT = deltaX/2.0;
x = linspace(a, b, nGridCells);
u0 = u0func(x);

tFinal = 2;
nTimeSteps = tFinal/deltaT;

%uExactFunc = @(x, t) -1.0*(x <= -t) + (x/t)*(x > -t && x < 0.5*t) + 0.5*(x >= 0.5*t);
uExactFunc = @(x, t) 1.0*(x <= 0.25*t) - 0.5*(x >= 0.25*t);
godunovSol = godunov(f, u0, deltaT, deltaX, nTimeSteps);
localLaxFriedrichsSol = localLaxFriedrichs(f, df, u0, deltaT, deltaX, nTimeSteps);

uExactSol = arrayfun(@(xj) uExactFunc(xj, tFinal), x);
plot(x, godunovSol(nTimeSteps+1,:), 'k:', x, localLaxFriedrichsSol(nTimeSteps+1, :), 'k
    ↪ --', x, uExactSol, 'k-', 'LineWidth', 2);
xlabel('x');
ylabel('u');
xlim([0.4, 0.6]);
%xlim([-2.5, 1.5]);
%ylim([-1.1, 0.6]);
title('T = 2');
legend('Godunov', 'Local Lax-Friedrichs', 'Exact', 'Location', 'southeast');
saveas(gcf, 'Figures/04_01.png', 'png');
```

The following images are produced. Both methods give good solutions, however the Godunov method is slightly less diffusive than the Local Lax-Friedrichs method.


T = 2
u
x
Godunov
Local Lax-Friedrichs
Exact


T = 2
u
x
Godunov
Local Lax-Friedrichs
Exact