Solution to the Inviscid Burgers' Equation by the Lax-Friedrichs Scheme*

Andre Gormann agormann@sfu.ca

Ethan MacDonald jem21@sfu.ca

1 Introduction

The problem we will be considering is the inviscid Burgers' equation with periodic boundary conditions

$$\begin{cases} u_t + uu_x = 0 \qquad x \in [-L, L] \qquad t \geq 0 \\ u(x,0) = u_0(x) \\ u(-L,t) = u(L,t) \end{cases} \tag{1}$$

Throughout the ensuing discussion, it will be far more useful if we rewrite the inviscid Burgers' equation in its conservation form

$$u_t + f(u)_x = 0 \qquad f(u) = \frac{1}{2}u^2 \tag{2}$$

We have chosen to use the Lax-Friedrichs scheme to approximate solutions of the Inviscid Burgers equation. The scheme is defined by the update rule

$$U_j^{n+1} = \frac{1}{2}\left(U_{j-1}^n + U_{j+1}^n\right) - \frac{\Delta t}{2\Delta x}\left(f(U_{j+1}^n) - f(U_{j-1}^n)\right)$$

where $U_j^n \in \mathbb{R}^N$ denotes the approximation to $u(x_j, t_n)$ at time t_n

The scheme combines upadtes in the spacial domain with time-stepping. Our reasoning for choosing the Lax-Friedrichs scheme is X Y Z. We will be using periodic boundary conditions because UMMMMMMMMMM WAVES.

*Placeholder title!

2 Theory

We will begin by introducing notation and then move on to discussion about the theory. First we discretize the interval $[-L, L]$ into a vector of N points x_j by defining a mesh width $\Delta x = 2L/N$ so that

$$x_j = -L + j\Delta x.$$

Note that we are assuming periodic boundary conditions so that $x_0 = x_N$, and so we have exactly N points. For reasons that will soon become clear, we are also interested in the half-steps $x_{j\pm 1/2}$ defined by

$$x_{j\pm 1/2} = x_j \pm \frac{\Delta x}{2}.$$

We will also denote the time step by Δt_n so that

$$t_n = n\Delta t_n.$$

The exact formula for Δt_n will remain undefined for now as it must be a variable time-step.

We denote the *pointwise values* of the true solution at the mesh point (x_j, t_n) by

$$u_j^n = u(x_j, t_n).$$

For finite difference methods, at each time step t_n we are computing a vector $U^n \in \mathbb{R}^N$ where the j-th component U_j^n approximates the true solution u_j^n. Specifically for conservation laws though, it is perhaps more natural to instead view U_j^n as approximating the *cell average* $\bar{u}_j^n$ about the mesh point (x_j, t_n) where

$$\bar{u}_j^n = \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n)dx. \tag{3}$$

The motivation for this interpretation comes from the integral form of the conservation law (2),

$$\frac{d}{dt}\int_{x_j-1/2}^{x_j+1/2} u(x,t)dx - \left[f(u(x_{j-1/2}, t)) - f(u(x_{j+1/2}, t))\right] = 0,$$

for which a derivation can be found in LeVeque, 1992, pp. 14-16. Integrating from t_n to t_{n+1} yields

$$\begin{aligned}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_{n+1})dx = &\int_{x_{j-1/2}}^{x_{j+1/2}} u(x, t_n)dx \\ &- \left[\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t))dx - \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t))dx\right].\end{aligned}$$

Now dividing by Δx and using the definition of (3) results in

$$\bar{u}_j^{n+1} = \bar{u}_j^n - \frac{1}{\Delta x}\left[\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t))dx - \int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t))dx\right]. \tag{4}$$

The goal of a successful numerical scheme then is to accurately model the flux through the boundaries of each cell. Explicitly, we want to find some numerical flux function $\mathcal{F}$ so that

$$\mathcal{F}(U_j^n, U_{j+1}^n) \approx \frac{1}{\Delta x}\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2}, t))dt \qquad \mathcal{F}(U_{j-1}^n, U_j^n) \approx \frac{1}{\Delta x}\int_{t_n}^{t_{n+1}} f(u(x_{j-1/2}, t))dt.$$

To this end, we say that a numerical method is in *conservation form* if

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left[\mathcal{F}(U_j^n, U_{j+1}^n) - \mathcal{F}(U_{j-1}^n, U_j^n)\right].$$

Note that while this derivation comes about through the introduction of control volumes, and hence falls under the umbrella of *finite volume methods*, we can also understand it through the lens of finite differences. Specifically, through the relation

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{F_{j+1/2}^n - F_{j-1/2}^n}{\Delta x} = 0$$

where $F_{j\pm1/2}^n \sim \mathcal{F}$ as before.

3 Body

4 Matrix Form of the Lax-Friedrichs Scheme

The Lax-Friedrichs scheme

$$U_j^{n+1} = \frac{1}{2}\left(U_{j-1}^n + U_{j+1}^n\right) - \frac{\Delta t}{2\Delta x}\left(f(U_{j+1}^n) - f(U_{j-1}^n)\right)$$

can be converted into a matrix form

$$\vec{U}^{n+1} = A\vec{U}^n - B\vec{f}(\vec{U}^n)$$

where

$$A = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 1 \\ 1 & 0 & 1 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 1 & 0 \end{bmatrix}, \quad B = \frac{\Delta t}{2\Delta x}\begin{bmatrix} 0 & 1 & 0 & \dots & 0 & -1 \\ -1 & 0 & 1 & \dots & 0 & 0 \\ 0 & -1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & -1 & 0 \end{bmatrix}$$

If we instead use the conservation form of the Lax-Friedrichs scheme

$$U_j^{n+1} = U_j^n - \frac{\Delta t}{\Delta x}\left(\mathcal{F}(U_j^n, U_{j+1}^n) - \mathcal{F}(U_{j-1}^n, U_j^n)\right)$$

$$\mathcal{F}(U_j^n, U_{j+1}^n) := \frac{\Delta x}{2\Delta t}(U_j^n - U_{j+1}^n) + \frac{1}{2}\left(f(U_j^n) + f(U_{j+1}^n)\right)$$

we get the following matrix form

$$\vec{U}^{n+1} = \vec{U}^n - C\vec{\mathcal{F}}(\vec{U}^n)$$

$$\vec{\mathcal{F}}(\vec{U}^n) = D\vec{U}^n + E\vec{f}(\vec{U}^n)$$

where

$$C = \frac{\Delta x}{\Delta t}\begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ 0 & 0 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 1 \\ 1 & 0 & 0 & \dots & 0 & -1 \end{bmatrix},$$

$$D = \frac{\Delta x}{2\Delta t}\begin{bmatrix} -1 & 0 & 0 & \dots & 0 & 1 \\ 1 & -1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -1 & 0 \\ 0 & 0 & 0 & \dots & 1 & -1 \end{bmatrix}, \quad E = \frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & 1 \end{bmatrix}$$

In either case, we can construct the matrices in MATLAB by using the *diag()* command on vectors containing the values of the non-zero diagonals, then filling in the values in the bottom left and top right corners as necessary, and lastly multiplying by the respective coefficient.

We will be using the matrices defined in this section to implement our periodic boundary conditions. The entries in the bottom left and top right corners of the matrices handle the cases where a value from beyond the periodic boundaries is needed.

5 Conclusion

References

Iserles, A. (2009). *A first course in the numerical analysis of differential equations*. Cambridge University Press.

LeVeque, R. J. (1992). *Numerical methods for conservation laws*. Birkhauser.

LeVeque, R. J. (2002). *Finite volume methods for hyperbolic problems*. Cambridge University Press.

Trefethen, L. N. (2000). *Spectral methods in matlab*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898719598