Numerical Analysis Project Report
A particle method for conservation laws

Aditya Kashi

December 23, 2016

Abstract

An elegant particle method for the solution of one-dimensional scalar conservation laws is presented, following the development by Farjoun and Seibold [2]. The method is based on particle 'management' based on conservation in the regions between the particles. Second-order accuracy is attained in both smooth regions and near shocks. The inviscid Burgers' equation is chosen here as the model problem to demonstrate the effectiveness of the method. However, extension to multi-dimensional problems and systems of conservation laws is not obvious.

1 Introduction

Background - numerical solution of conservation laws

State-of-the-practice (engineering): finite volume or finite difference methods with limiters, ENO/WENO or artificial dissipation terms.

- second-order accurate only in smooth regions
- between first and second-order accurate at shocks
- TVD can usually be guaranteed only for scalar case

Based on 'Eulerian' method, usually.

Solve scalar 1D conservation laws with a second-order TVD scheme.

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = 0 \tag{1}$$
$$u(x,0) = u_0(x) \tag{2}$$

Burgers' equation: $f(u) = \frac{u^2}{2}$.

2 The method

The characteristic equations are key. [2]

$$\dot{x} = f'(u) \tag{3}$$

$$\dot{u} = 0 \tag{4}$$

The solution is a curve along which u is constant, while u is smooth. We can think in terms of 'particles'.

Conservative particle management and interpolation

'Particle management' is needed when particles collide or get too far apart.

- Based on local conservation of total 'amount of u' between two particles.
- Rate of change of amount of u contained between two particles

$$\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} u(x,t)dx \tag{5}$$

- However, care must be taken because the limits of integration are now time-dependent.

After some calculation it can be shown, for space-independent flux functions f, that

$$\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} u(x,t)dx = (x_2(t) - x_1(t))a_f(u_1, u_2) \tag{6}$$

where

$$a_f(u_1, u_2) = \frac{[f'(u) - f(u)]_{u_1}^{u_2}}{[f'(u)]_{u_1}^{u_2}}. \tag{7}$$

For Burgers' equation is

$$a_f(u_1, u_2) = \frac{u_1 + u_2}{2} \tag{8}$$

This is the basis of conservative particle management and conservative interpolation.

It has been shown that the method is second-order accurate away from shocks and TVD.

A 'shock location' step has to be performed to achieve second-order accuracy in presence of shocks. However, for Burgers' equation, second order accuracy seems to be achieved irrespective of this.

2.1 Implementation

- Since particles may need to be added or deleted anywhere, an array is not a good data structure. A linked list was coded, which allows efficient addition and deletion, at the cost of more expensive iteration through the particles.

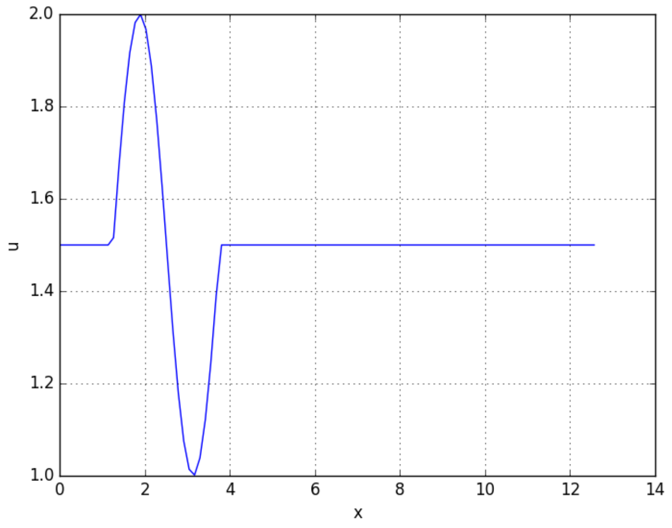

Figure 1: Initial condition

- Conservative particle management is cheap only for quadratic flux functions. For others, a Newton iteration is required to solve for the new particles.
- Implementation in Julia programming language [1]

3 Some results

4 Conclusions and future work

- The particle method provides a way to achieve second-order accuracy for non-smooth solutions, at least for scalar conservation laws.
- An easy post-processing step is needed to maintain second-order accuracy in presence of shocks.
- However, extension to multi-dimensional problems and systems of conservation laws is not straightforward.

4.1 Future work

4.1.1 Systems of conservation laws

- $\boldsymbol{f}'(\boldsymbol{u})$ is now a matrix with N eigenvalues

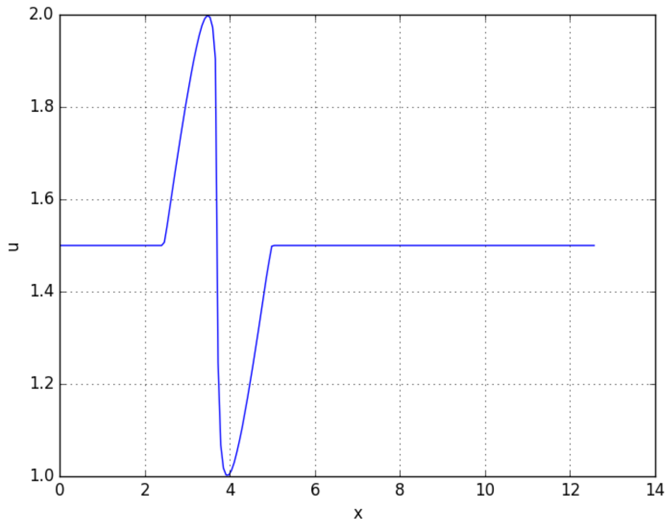

Figure 2: t = 0.1, before shock formation

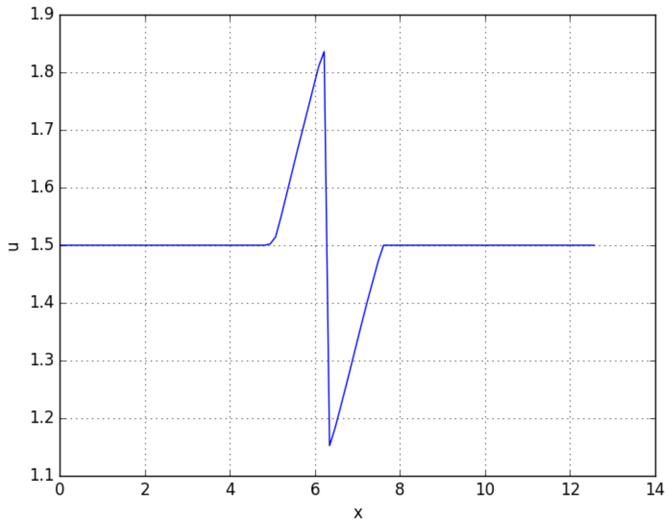

Figure 3: t = 2.5, after shock formation

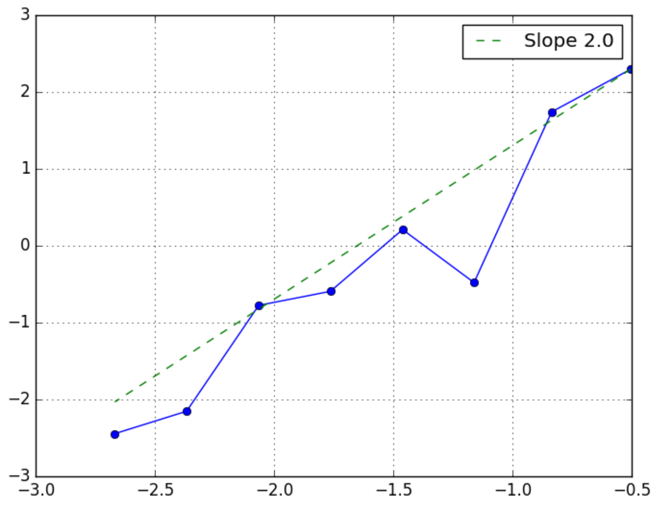

Figure 4: Grid convergence for t = 2.5. The x axis is $\log_{10}||h||$, the y axis is $\log_{10}||u - u_{ref}||$

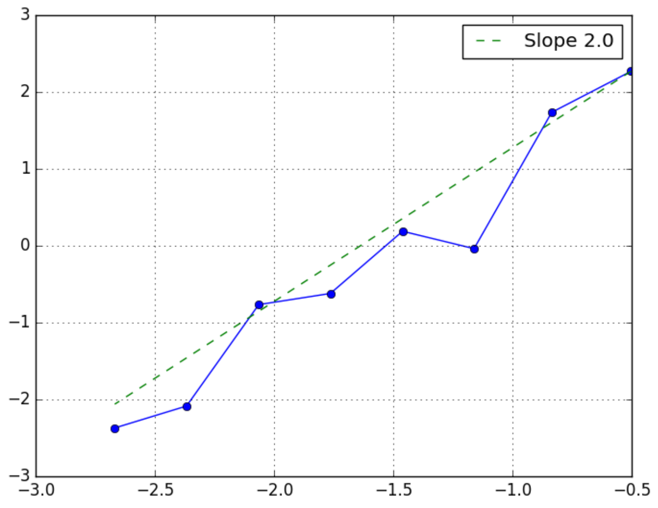

Figure 5: t = 2.5; no post-processing

- Speed of the particle?
- One possible solution: use N different sets of particles.
- Can TVD still be guaranteed?

4.1.2 Multidimensional problems

- In 1D, we have ‘next’ and ‘previous’ particles. Not so in multi-D.
- One could compute a Voronoi tessellation to get ‘neighbors’ for each particle, and apply the 1D merge/insert for each pair thus generated.
- Implementation - data structures?

4.1.3 Others

- Steady-state solutions?
- Implicit time stepping?

References

[1] Jeff Bezanson et al. “Julia: A Fast Dynamic Language for Technical Computing”. In: *CoRR* abs/1209.5145 (2012). URL: `http://arxiv.org/abs/1209.5145`.

[2] Y. Farjoun and B. Seibold. “An exactly conservative particle method for one-dimensional scalar conservation laws”. In: *Journal of Computational Physics* 228.14 (2009), pp. 5298–5315.