Flux-Conservative Hermite Methods for Simulation of Nonlinear Conservation Laws

Adeline Kornelus1 · Daniel Appelö2

Received: 16 March 2017 / Revised: 8 September 2017 / Accepted: 15 November 2017 / Published online: 29 November 2017
© Springer Science+Business Media, LLC, part of Springer Nature 2017

Abstract A new class of Hermite methods for solving nonlinear conservation laws is presented. While preserving the high order spatial accuracy for smooth solutions in the existing Hermite methods, the new methods come with better stability properties. Artificial viscosity in the form of the entropy viscosity method is added to capture shocks.

Keywords Hermite · Conservative · High order method · The entropy viscosity method

1 Introduction

Conservation laws model physical systems arising in traffic flows, aircraft design, weather forecast, and fluid dynamics. Numerical methods for conservation laws ideally conserve quantities like mass or energy, and accurately capture various physical components of the solutions, from small smooth scales to shock waves. The presence of both smooth waves and shock waves, for example in shock-turbulence interaction creates a challenge to the simulation of nonlinear conservation laws.

Shock waves typically appear in solutions to nonlinear conservation laws, and are characterized by very thin regions where the solution changes rapidly. Approximation of shocks
and small waves is challenging as the small and large scales need to be resolved simultaneously. Historically, low order finite volume and finite difference methods equipped with flux/slope limiters have been used to handle shocks, see for example the textbooks [16,17]. The drawback with low order methods is that they cannot accurately propagate small scales over long distances and as a result, today the research focus has gravitated towards high order accurate methods with shock capturing capability.

Among high order methods, the weighted essentially non-oscillatory (WENO) method [12,18,22], has proven to be a method popular among practitioners. WENO methods are still relatively dissipative which may be a drawback for turbulent simulations [19]. Also, discontinuous Galerkin methods combined with shock capturing [6,15], or selectively added artificial viscosity [14,20,27], have received significant interest. The latter approach traces back to the artificial viscosity method by Neumann and Richtmyer [25] and the popular streamline diffusion method [4,13], which computes the viscosity based on the residual of the PDE.

In this work, we advocate the combination of a high order method and selectively added artificial viscosity. Specifically, we show how the entropy viscosity by Guermond and Pasquetti [9], can be implemented in our new flux-conservative Hermite methods.

First introduced by Goodrich et al. [7] for hyperbolic initial-boundary value problems, Hermite methods use the solution and its first \(m\) derivatives in each coordinate to construct an approximate solution to the PDE. In the flux-conservative Hermite methods, the numerical flux is computed at cell edges and then interpolated to cell center for time evolution, hence by construction, the numerical flux is continuous at cell interface. Additionally, for nonlinear problems, it is more efficient to use one step methods than the Taylor series approach in [7], see [10,11]. Here we will use the standard Runge–Kutta method to evolve in time.

The rest of the chapter is organized as follows. In Sect. 2, we derive conservation laws and discrete conservation. Then, in Sect. 3, we describe Hermite methods as first introduced by Goodrich et al. [7], followed by the description of the flux-conservative Hermite methods in Sect. 3.2. For shock capturing capability, we implement the entropy viscosity method, which is explained in Sect. 4. In Sect. 5, we present the results of simulation of Euler’s equations, see [1] for results on Burgers’ equation.

2 Conservation Laws

A scalar conservation law in one space dimension takes the form

\[
\frac{\partial}{\partial t} u(x, t) + \frac{\partial}{\partial x} f(u(x, t)) = 0, \tag{1}
\]

where \(u(x, t)\) is the state variable at location \(x\) and time \(t\) and \(f(u)\) is the flux, or the rate of flow, of the state variable \(u\).

The derivation of conservation laws comes from the observation that at any given time \(t\), the rate of change of the total quantity of the state variable \(u\) over some interval \([a, b]\) must be equal to the net flux \(f(u)\) into the interval through the endpoints. Mathematically, this can be expressed as
\[
\frac{d}{dt} \int_a^b u \, dx = f(u(a, t)) - f(u(b, t)).
\]  

(2)

When approximating the solution to scalar conservation laws given by Eq. (1), the PDE is typically discretized on a grid consisting of \(N_x\) cells where \(x_0 = a\) and \(x_{N_x} = b\). It is desirable that the numerical method satisfies discrete conservation. If the reconstructed solution \(u^j_h\) and flux \(f^j_h\) on any cell \(I_j = [x_{j-1}, x_j]\) satisfy the condition \(f^j_h|_{x=x_{j-1}} = f^{j+1}_h|_{x=x_j}, j = 1, \ldots, N_x - 1,\) we immediately find

\[
\int_a^b \frac{\partial u^j_h}{\partial t} \, dx = \sum_{j=1}^{N_x} \int_{x_{j-1}}^{x_j} \frac{\partial u^j_h}{\partial t} \, dx
\]

\[
= \sum_{j=1}^{N_x} \int_{x_{j-1}}^{x_j} \frac{\partial}{\partial x} (-f^j_h) \, dx
\]

\[
= \sum_{j=1}^{N_x} \left( f^j_h|_{x=x_{j-1}} - f^j_h|_{x=x_j} \right)
\]

\[
= f^j_h(u(a)) - f^j_h(u(b)).
\]  

(3)

The property that \(f^j_h|_{x=x_{j-1}} = f^{j+1}_h|_{x=x_j}\) does not hold for the original Hermite methods, and our goal here is to design a new Hermite method with this property. Before describing our new method, we briefly describe the original method.

3 Hermite Methods

A Hermite method of order \(2m + 1\) approximates the solution to a PDE by an element-wise polynomial that has continuous spatial derivatives up to order \(m\) at the element’s interfaces. In Hermite methods, the degrees of freedom are function and spatial derivative values, or equivalently the coefficients of the Taylor polynomial at the cell center of each element. The evolution of the degrees of freedom on each element can be performed locally.

3.1 Hermite Method in One Dimension

Consider again Eq. (1) on the domain \(D = [x_L, x_R]\). Let \(G^p\) and \(G^d\) be the primal grid and the dual grid, defined as

\[
G^p = \{ x_j = x_L + jh_x, \, j = 0, \ldots, N_x \},
\]

(4)

\[
G^d = \{ x_{j+1/2} = x_L + \left( j + \frac{1}{2} \right) h_x, \, j = 0, \ldots, N_x - 1 \},
\]

(5)

where \(h_x = (x_R - x_L)/N_x\) is the distance between two adjacent nodes. Let \(u^p\) and \(u^d\) be the approximations to the solution on the primal and dual grids, respectively.

At the initial time \(t_n = t_0 + n \Delta t\), we assume that the solution and its derivatives \((u^p, (u^p)', (u^p)'', \ldots, (u^p)^{(m)})\) are given on the primal grid. Starting from time \(t = t_n\) on the primal grid \(G^p\), we evolve the solution one full time step by:
Reconstruction by Hermite interpolation We construct \( u^d \), the global Hermite interpolation polynomial of degree \((2m + 1)\), on the dual grid. That is,

\[
u^d(x, t_n) = \sum_{k=0}^{2m+1} b_k(t_n) \left( x - x_{j-1/2} \right)^k, \quad x \in \mathcal{I}^p_j = [x_{j-1}, x_j],
\]

where the coefficients \( b_k(t_n) \) are uniquely determined by the interpolation conditions

\[
\frac{\partial^k}{\partial x^k} u^d(x_i, t_n) = \frac{d^k}{dx^k} u^p(x_i, t_n), \quad k = 0, \ldots, m, \quad i = j - 1, j.
\]

Time evolution By rewriting Eq. (1) as \( u_t = -f(u)_x \), it is obvious that in order to evolve \( u^d \), we need to compute a polynomial approximation \( f^d \) to the flux function \( f(u) \). We offer two ways to obtain \( f^d \):

- **Modal approach:** For many nonlinear flux functions it is possible to perform polynomial operations on \( u^d \) to obtain a resulting truncated polynomial approximating, see [11].

- **Pseudospectral approach:** Compute a local polynomial \( f_h^s \) interpolating \( f(u^d) \) on a quadrature grid \( \mathcal{G}^{ps} \) inside \( \mathcal{I}^p_j, j = 1, \ldots, N_x \), and transform \( f_h^s \) to a Taylor polynomial \( f^d \) of suitable degree.

We differentiate \( f^d \) in polynomial sense to get an approximation to the derivative of the flux function \( f(u)_t \). We usually use the modal approach unless this option is not applicable. Next, we derive an ODE to evolve \( u^d \), or equivalently the coefficients of the Hermite interpolant \( \mathbf{c}(t) = (c_0(t), \ldots, c_{2m+1}(t))^T \), by insisting that the numerical solution \( u^d \) satisfy Eq. (1) along with derivatives of (1), at the cell centers \( x = x_{j-1/2}, j = 1, \ldots, N_x \). The resulting system of ODE for \( c_k, k = 0, \ldots, 2m + 1 \), can then be evolved independently on each \( \mathcal{I}^p_j \) with any ODE solver. The reconstruction step provides the initial data, \( \mathbf{c}(t_n) \).

Repeat on dual grid At the end of the half time step, we have evolved the degree \( 2m + 1 \) polynomial \( u^d \) from time \( t = t_n \) to \( t = t_{n+1/2} \). Before repeating the above process, we truncate \( u^d(x, t_{n+1/2}) \) by removing the coefficients \( c_k, k = m + 1, \ldots, 2m + 1 \) and extracting the solution and its derivatives \( u^d, (u^d)_t, (u^d)_x, \ldots, (u^d)_x^{(m)} \) on the dual grid using the Taylor series formula \( (u^d)_x^{(k)} = k! c_k \). We then repeat the above process (including the truncation) to obtain \( u^p \) at time \( t = t_{n+1} \), see Fig. 1 for illustration.

3.1.1 Example: Burgers’ Equation

To illustrate the specifics of the time evolution, we consider Burgers’ equation, with \( f(u) = u^2/2 \), approximated by \( f_h = u_h^2/2 \), where \( u_h \) represents the degree \((2m + 1)\) interpolant on either of the grids. We can write

\[
(u_h)_t + (f_h)_x = 0, \\
(u_h)_t + (f_h)_xx = 0, \\
(u_h)_t + (f_h)_xxx = 0, \\
\vdots \\
\]

where

\[
f_h = \sum_{k=0}^{2m+1} b_k(t)(x - x_{j-1/2})^k.
\]
Fig. 1 Illustration of the numerical process in one dimensional Hermite methods for a full time step. Solid circles represent the primal grid $G^p$ and open circles represent the dual grid $G^d$. $\mathcal{I}$ is the Hermite interpolation operator and $\mathcal{T}$ is the time evolution operator.

The coefficients $b_k$ are determined by truncated polynomial multiplication, that is $b_k = \frac{1}{2} \sum_{l=0}^{k} c_l c_{k-l}$. Insisting that the approximate solution $u_h$ satisfy Eq. (8) at the cell centers $x = x_{j-1/2}$, we obtain

$$\begin{bmatrix}
c'_0(t) \\
c'_1(t) \\
\vdots \\
c'_{2m}(t) \\
c'_{2m+1}(t)
\end{bmatrix} = \begin{bmatrix}
b_1(t) \\
2 b_2(t) \\
\vdots \\
(2m + 1) b_{2m+1}(t) \\
0
\end{bmatrix}.$$  \hspace{1cm} (9)

While Eq. (9) is valid for any cell, the initial data for each cell are different from one cell to another. For a more detailed explanation and open source implementations, see [3,10,11].

3.2 Flux-Conservative Hermite Methods

The numerical flux $f_h$ obtained by the approach described above, is discontinuous at cell interfaces when the flux function $f(u)$ is nonlinear. Numerically, the discontinuity in the flux induces numerical stiffness. As a result, the time step often needs to be taken smaller than it needs to be. To remedy this, we propose new flux-conservative Hermite methods that impose flux continuity by computing the numerical flux at cell edges, and then interpolate the numerical flux to cell center.

To illustrate the difference between the original and flux-conservative Hermite schemes, we plot the numerical flux $f_h = u_h^2/2$ with $m = 3$ and for $N_e = 3$ elements in Fig. 2. The numerical flux obtained using the original Hermite method, shown as the blue curve, has discontinuities at cell interfaces. On the other hand, the numerical flux obtained by the flux-conservative Hermite method, shown as the black curve, is continuous everywhere.

3.2.1 The Construction of the New Method

The goal of our construction is to globally conserve the integral of $u_h$ and its $m$ first derivatives over one half time step, with $\Delta t = \Delta t/2$.

In the flux-conservative methods, we assume that the solution on the primal grid at time $t_n$ is given by
Fig. 2 Numerical flux for Burgers’ equation with random data. Here, the domain is divided into three elements and the numerical flux is computed using the original Hermite (blue) and the flux-conservative Hermite (black). Notice that the numerical flux obtained by the original Hermite method is discontinuous at the cell interface, while the numerical flux obtained by the flux-conservative Hermite method is continuous everywhere (Color figure online)

\[ u^p(x, t_n) = \sum_{k=0}^{2m+1} c_k(t_n) (x - x_j)^k, \quad x \in T^d_j = [x_{j-1/2}, x_{j+1/2}]. \]  

(10)

Note that the degree of this polynomial is different than in the original method. We assume that the time stepping will be performed by an explicit one step method requiring stage values. The evolution will be carried out at the cell center where the stage values will be the derivative of the Hermite interpolant of the flux. As this interpolant is \( m \) times differentiable at the edges, it will result in a conservative discretization. Now, the time evolution of the approximate solution entails the following steps.

**Computation of the stage fluxes at the cell edges** For simplicity, assume that we use the classic fourth order Runge–Kutta, then to construct the Hermite interpolants for the four stages we first compute

\[
F_1^p = \mathcal{F}(u^p), \\
F_2^p = \mathcal{F}(u^p + \frac{\Delta t}{2} \frac{dF_1^p}{dx}), \\
F_3^p = \mathcal{F}(u^p + \frac{\Delta t}{2} \frac{dF_2^p}{dx}), \\
F_4^p = \mathcal{F}(u^p + \Delta t \frac{dF_3^p}{dx}),
\]

where \( \mathcal{F}(*) \) is an operator that computes a Taylor polynomial \( \tilde{f} \) of degree \( 2m + 1 \) approximating the flux \( f(*) \), either by modal or pseudospectral approach (see Sect. 3.1). In the modal approach, the arguments of \( \mathcal{F} \) are substituted into the flux function \( f \) to obtain the numerical flux \( \tilde{f} \). Due to the nature of nonlinearity in the flux function, the resulting polynomial \( \tilde{f} \) is typically a higher degree polynomial than the degree of the input argument. Hence the operator \( \mathcal{F}(*) \) truncates such polynomials to the same degree as the input argument.

**Reconstruction of solution and fluxes by Hermite interpolation** Next, we construct \( u^d \) and \( F_i^d, i = 1, \ldots, 4 \), the global Hermite interpolation polynomials of degree \( (2m + 1) \) of the solution and the flux, respectively. Let \( w^d \) represent \( u^d \) or \( F_i^d \) and \( w^p \) represent \( u^p \) or \( F_i^p \). Then,
Theorem 1
Assume we use the flux-conservative Hermite method to evolve \( u_t + f(u)_x = 0 \) with periodic boundary conditions. Further assume that \( ud(t, x) \) is the periodic degree \( 2m+1 \) Hermite interpolating polynomial and that \( F_i^d, i = 1, \ldots, s \) are the periodic degree \( 2m+1 \) polynomials Hermite interpolating the fluxes. Further, let the coefficients \( c_k(t) \) of \( ud \) on a cell be evolved from time \( t = t_n \) to \( t = t_n + \Delta \hat{t} \) by the one step method

\[ c_k(t_n + \Delta \hat{t}) = c_k(t_n) + \Delta \hat{t} \sum_{i=1}^{s} \alpha_i (k + 1)b_{k+1}^{(i)}(t_n) = 0, \quad k = 0, \ldots, 2m+1, \]

where \( b_{k}^{(i)} \) are the coefficients of \( F_i^d \). Then, the following conservation statement holds:

\[ \sum_{j} \int_{x_{j-1}}^{x_j} \frac{\partial}{\partial x^k} ud(t_n + \Delta \hat{t}, x) \, dx = \sum_{j} \int_{x_{j-1}}^{x_j} \frac{\partial}{\partial x^k} ud(t_n, x) \, dx, \]

\[ k = 0, \ldots, 2m + 2 - s. \] (13)

Proof Using the formula for a one-step time stepping method for the conservation law (1)

\[ \frac{ud(t_{n+1/2}) - ud(t_n)}{\Delta \hat{t}} = \Delta \hat{t} \sum_i \alpha_i \frac{dF_i^d}{dx}, \] (14)
Fig. 3 Original versus flux conservative Hermite methods. Here, we dropped the subscript $h$ in all the computed quantities for compactness together with the continuity of $m$ first derivatives of each of the $F^d_i$, the result follows immediately from the update formula. Note also that $F^d_{i+1}$ is one order less accurate than $F^d_i$ due to flux differentiation during stage $i$.

To summarize, in the original Hermite scheme, computation of numerical fluxes is performed at the cell center using the interpolated solution. The flux-conservative Hermite scheme requires a computation (and storage) of numerical fluxes at the cell edges and the interpolation of those fluxes to the cell center. Refer to Fig. 3 for an illustrative comparison between the schemes.

3.3 The Flux-Conservative Hermite Method in Two Dimensions

Now, let us consider a conservation law

$$u_t + (f(u))_x + (g(u))_y = 0,$$  \hspace{1cm} (15)

on the domain $D = [x_L, x_R] \times [y_B, y_T]$. Let $G^p$ and $G^d$ be the primal and dual grids, defined as

$$G^p = \{(x_i, y_j) \} = (x_L + i h_x, y_B + j h_y), \quad i = 0, \ldots, N_x, \quad j = 0, \ldots, N_y,$$  \hspace{1cm} (16)

$$G^d = \{(x_{i+1/2}, y_{j+1/2}) = (x_L + \left( i + \frac{1}{2} \right) h_x, y_B + \left( j + \frac{1}{2} \right) h_y) \}, \quad i = 0, \ldots, N_x - 1, \quad j = 0, \ldots, N_y - 1,$$  \hspace{1cm} (17)

where $h_x = (x_R - x_L)/N_x$ and $h_y = (y_T - y_B)/N_y$ are the distances between two adjacent nodes in $x$ and $y$ directions respectively.

The extension of the flux-conservative method from one dimension is straightforward. Writing Eq. (15) as $u_t = -f(u)_x - g(u)_y$ and letting $u^d(t)$ represent the two-dimensional tensor product Hermite interpolant of the data on the primal grid we can write the RK4 time stepping of $u^d(t_n)$ to time $t = t_n + \Delta t$ as

$$\frac{u^d(t_n + \Delta t) - u^d(t_n)}{\Delta t} = \frac{K_1^d + 2K_2^d + 2K_3^d + K_4^d}{6}.$$  \hspace{1cm} (18)
Table 1  Comparison of costs in original and flux-conservative Hermite methods, \( n_K \) is the number of stages in Runge–Kutta method, \( d \) is the spatial dimension

|                | Flux computation      | Interpolation         |
|----------------|-----------------------|-----------------------|
| Original       | \( n_K d(2m + 2)^2d \) | \( 2^{2d-1}(2m + 2)^{d+1} \) |
| Flux-conservative | \( n_K d(2m + 2)^2d \) | \( (1 + n_K d)2^{2d-1}(2m + 2)^{d+1} \) |

The left hand side of Eq. (18) is an approximation to \( u_t \) and the right hand side is an approximation to stage values of \(- (f(u)_x + g(u)_y)\). Similar to the one dimensional case we have

\[
K^d_1 = -(F^d_1)_x - (G^d_1)_y, \\
K^d_2 = -(F^d_2)_x - (G^d_2)_y, \\
K^d_3 = -(F^d_3)_x - (G^d_3)_y, \\
K^d_4 = -(F^d_4)_x - (G^d_4)_y.
\]

Here, for example, \( F^d_i \) is the degree \( 2m + 1 \) tensor product polynomial that interpolates \( f(u^p + \gamma_i \Delta \hat{t} F^p_i) \) and its \( m \) first derivatives in \( x \) and \( y \) at the four adjacent primal grid-points.

3.4 Comparison of Computational Costs

The time evolution portion of the Hermite methods are performed by a one step method with \( n_K \) stages, involving computation of the flux function, interpolation of the solution and, in the flux-conservative method, the fluxes, and differentiation of fluxes. For the purpose of this comparison, we assume Burgers’ flux function \( f(u) = u^2/2 \) in 1D or \( f(u) = g(u) = u^2/2 \) in 2D. Each 1-dimensional Hermite interpolation is equivalent to a multiplication by a \((2m + 2)\) by \((2m + 2)\) matrix. If we use the recipe above, each 2-dimensional Hermite interpolation corresponds to \( 2 \times (2m + 2) \) one-dimensional interpolations. The factor \((2m + 2)\) comes from the the fact that the \( y \) dimension brings in \((2m + 2)\) interpolations in 1D and the multiplicative factor 2 comes from the fact that we interpolate in \( y \) direction on both the left and right edges of the cell. In 3D, we have four interpolations in the \( z \) direction, each brings in \((2m + 2)\) times interpolations in 2D, and so on. We summarize the cost of the method, corresponding to the number of multiplications involved, in Table 1. The number of interpolation in the flux-conservative Hermite method is \( n_K d \ 2^{2d-1} \) more than the original Hermite method. We note that the flux-conservative scheme can be simplified to just two flux interpolations by adding up the \( F \)'s and \( G \)'s, but in this chapter, we interpolate each flux separately. There is also an additional cost of differentiation at cell corners, but it is negligible compared to the cost of interpolation.

4 The Entropy Viscosity Method

Given a PDE of the form (1), there exists an entropy function \( E(u) \) and its corresponding entropy flux function \( F(u) = \int E'(u)f'(u) \) such that the entropy residual satisfies

\[
r_{EV}(u) \equiv E_t(u) + \nabla \cdot F(u) \leq 0.
\]
This inequality can be used to select the physically correct solution to (1) or (15). The direction of the inequality can vary from one problem to another, but the residual takes a nonzero value only at shocks. In essence, the entropy viscosity (EV) method uses the fact that the residual approaches a Dirac delta function centered at shocks to construct a nonlinear dissipation. The resulting dissipation is small away from shocks and has a sufficient amount to regularize the shocks. The details of EV for conservation laws are described in detail in [8] but we summarize its most important features here.

Consider the conservation law whose right hand side has been replaced by a viscous term, \( u_t + \nabla \cdot f(u) = \nabla \cdot (v \nabla u) \), with \( v = v_h(x, t) \) given by

\[
 v_h(x, t) := \min(v_{EV}, v_{max}),
\]

where \( v_{EV} \) is the entropy-based viscosity and \( v_{max} \) is a viscosity whose size depends on the largest eigenvalue of the flux function \( f(u) \), representing the maximum wave speed. The discretized entropy-based viscosity \( v_{EV} \) is then given by

\[
 v_{EV}(x, t) = \alpha_{EV} C_1(u_h) h^\beta |r_{EV}(u_h)|,
\]

where \( \beta \) is a positive scalar, \( \alpha_{EV} \) is a user defined constant and \( C_1(u_h) \) is some PDE-specific normalization.

At shocks, the entropy residual approaches a Dirac delta function, so we instead use

\[
 v_{max}(x, t) = \alpha_{max} h \max_{y \in V_x} C_2(u_h(y, t)),
\]

where \( \alpha_{max} \) is another user defined constant, \( C_2 \) serves as the maximum wave speed and \( V_x \) is some neighborhood of \( x \). The \( V_x \) neighborhood can either be “local”, i.e. containing only a few cells around \( x \), or “global”, i.e. \( V_x = D \), where \( D \) is the domain of the PDE. In this work, we use global \( V_x \).

In recent papers, the parameter \( \beta \) is chosen to be 2, but we found that this may prevent convergence for moving shocks, see [1] where we also argue that \( \beta = 1 \) is a more appropriate choice. In essence our argument is as follows. As the entropy residual approaches a Dirac delta distribution, a consistent discretization of the residual with a single shock must satisfy

\[
 h_j (r_{EV})_j = 1,
\]

where \( h_j = x_{j+1} - x_j \). Thus, we expect \( (r_{EV})_j \sim h_j^{-1} \) near the shock. When \( \beta = 2 \), the two terms \( v_{EV} \) and \( v_{max} \) are both \( \mathcal{O}(h) \). Since the parameters are tuned on a coarse grid and the terms \( C_1 \) and \( C_2 \) in (21)–(22) also change with the grid size, the selection of the minimum of \( v_{EV} \) and \( v_{max} \) does not necessarily “converge” as the grid gets refined. If instead we choose \( \beta = 1 \), then \( v_{EV} = \mathcal{O}(1) \) while \( v_{max} = \mathcal{O}(h_j) \), and the particular choice of \( \alpha_{EV} \) is thus irrelevant in the limit \( h_j \rightarrow 0 \) as the selection mechanism will eventually select \( v_{max} \) at the shocks.

While the explicit formula for \( C_1 \) and \( C_2 \) varies from one PDE to another, the core of the entropy viscosity method remains the same. The size of the entropy residual gives us a sense of relative distance to the shock, which is then used to take the following actions: near a shock, EV uses sufficiently large dissipation, \( v_h = v_{max} \), and away from a shock, EV uses entropy-based dissipation, \( v_h = v_{EV} \).
5 Numerical Examples

We start by demonstrating the need for the flux-conservative Hermite method in the case of Burgers’ equation with a moving shock, and then we verify that the rates of convergence for a smooth solution, $2m + 1$, (in space) are the same for the new flux-conservative method as for the original method. Furthermore, when the entropy viscosity is discretized accurately enough, we recover the $2m + 1$ rate of convergence. The parameters for Burgers’ simulation can be found in Table 2.

**Tuning EV parameters** We perform the following steps on a coarse grid to save computational cost. The first goal is to ensure enough viscosity in order to resolve shocks. Therefore, we tune $\alpha_{\text{max}}$ until we get a smooth solution while setting $\alpha_{\text{EV}}$ to “infinity.” The next goal is to get a better resolution away from shocks. We then fix $\alpha_{\text{max}}$ and reduce $\alpha_{\text{EV}}$ as much as possible without getting overshoots/undershoots.

5.1 Original Versus Flux-Conservative Hermite Methods

We solve the viscous Burgers’ equation

$$u_t + \left(\frac{u^2}{2}\right)_x = \nu u_{xx}$$

(24)

on $[-1, 1]$ with initial data

$$u_0(x) = \begin{cases} 0.6 & x < -0.5, \\ -0.4 & x > -0.5, \end{cases}$$

(25)

and fixed boundary condition $u(\pm 1, t) = u_0(\pm 1)$. The initial data corresponds to a shock moving with speed $v_s = 0.1$. We compute the solution up to time $t = 5$. The solution at the final time is simply a translation of the initial data by distance $\Delta x = 0.5$.

The viscosity coefficient $\nu$ is determined by the entropy viscosity method. The discretized viscosity coefficient $\nu_h$ is given by the formula

$$\nu_h = \min(\nu_{\text{max}}, \nu_{\text{EV}}),$$

(26)

$$\nu_{\text{EV}} = \alpha_{\text{EV}} h_x |r_{\text{EV}}(x, t)| |\Delta u_h|,$$

(27)

$$\nu_{\text{max}} = \alpha h_x \max_{y \in D} |u_h(y, t)|,$$

(28)

where $h_x$ is the grid size, $u_h$ is the approximation to the solution $u$ of the PDE, $\Delta u_h$ is the increment in $u_h$, i.e. $\Delta(u_h)_j = (u_h)_{j+1} - (u_h)_j$, and $r_{\text{EV}}$ is the discretized residual of the entropy inequality using the entropy function $E = u^2/2$, and the associated entropy flux function $F = u^3/3$,

$$r_{\text{EV}} = \partial_t (u_h^2/2) + \partial_x (u_h^3/3).$$
We use a low order accurate discretization since we are primarily interested in the convergence of the solution, instead of the accuracy. We discretize the temporal derivative of the flux function \( \partial_t E_h = \partial_t (u_h^2/2) \) using the 2nd order backwards differentiation formula:

\[
\partial_t E^n_j = \frac{3E^n_j - 4E^{n-1}_j + E^{n-2}_j}{2\Delta t}.
\]

The discretization of the spatial derivative of the entropy flux function \( \partial_x F_h = \partial_x (u_h^3/3) \) is performed using the centered Finite Difference

\[
\partial_x F^n_j = \frac{F^n_{j+1} - F^n_{j-1}}{2h_x}.
\]

Notice that in the original Hermite method, the parameter \( \alpha_{EV} \) appearing in the viscosity \( \nu_{EV} \) is intentionally set to be large so that the other viscosity \( \nu_{\text{max}} \) is applied in a wider region near shock and therefore resolving the shock, see Eq. (20).

In Fig. 4, we display the numerical solutions obtained with a sequence of grid sizes. On the left, the solutions are computed using a original Hermite method, and on the right, the solutions are computed using a flux-conservative Hermite method. The x-axis has been truncated for better presentation of the difference between the two methods. Notice that the solutions obtained with the original Hermite method converge to an incorrect solution (corresponding to wrong shock speed), while the solutions obtained with the flux-conservative Hermite method converge to the physically correct solution. We were not able to run the original Hermite method at the same CFL as the flux-conservative Hermite method.

### 5.2 Convergence for a Smooth Solution

We solve Burgers’ equation

\[
u + \left( \frac{u^2}{2} \right)_x = 0,
\]

on the domain \( x \in [-\pi, \pi] \) with periodic boundary conditions. The initial data is \( u(x, 0) = -\sin(x) + 0.3 \) and we evolve the solution until time \( t = 0.4 \). The timestep is chosen as \( \Delta t = \text{CFL} h_x / \max_x |u(x, 0)| \), with CFL = 0.1.

In Table 3, we display the maximum error at the final time computed against a reference solution obtained using \( m = 5 \) and \( h_x = \pi/64 \). We can see from the table that the rates of convergence are consistent with the predicted rate \( 2m + 1 \).
We then repeat the experiment but we now add a viscous term,
\[ u_t + \left( \frac{u^2}{2} \right)_x = \nu u_{xx}. \]  
(30)

The viscosity coefficient \( \nu \) is determined by the entropy viscosity method. The discretized viscosity coefficient \( \nu_h \) is given by Eq. (26) in an earlier experiment.

In order to maintain high rate of convergence, we discretize the temporal derivative of the flux function \( \partial_t E_h = \partial_t \left( \frac{u^2 h}{2} \right) \) using the 5th order backwards differentiation formula:
\[
\partial_t E^n_j = \frac{137E^n_j - 300E^{n-1}_j + 300E^{n-2}_j - 200E^{n-3}_j + 75E^{n-4}_j - 12E^{n-5}_j}{60 \Delta t}.
\]

To ensure sufficient accuracy in the discretization of the entropy residual, we discretize the spatial derivative of the entropy flux function \( \partial_x F_h = \partial_x (u_{h}^3/3) \) using a pseudospectral approach. The term \( |\Delta u_h| \) is proportional to \( h_x \) when \( u_h \in C^1([-\pi, \pi]) \). Hence, the term \( \nu_{EV} \) is proportional to \( h_x^7 \), which means that the approximation for \( \nu_{EV} \) is at least as accurate as the Hermite method itself. Comparing the convergence results in Tables 3 and 4, we observe that when the grid size is small enough, the error in the solution to the viscous PDE is comparable to the error in the solution to the inviscid PDE.

We next present a sequence of experiments displaying the performance of the Hermite–Runge–Kutta-4-Entropy-Viscosity method for Euler’s equations (with artificial viscosity).

### 5.3 Euler’s Equations in One Dimension

We consider Euler’s equations which represent conservation of mass, momentum, and energy,
Here, $\rho$ is the mass density, $\rho u$ is the momentum density, $u$ is the velocity, and $E$ is the internal energy per unit mass. Furthermore, we assume an ideal gas with the equation of state

$$E = \frac{p}{\gamma - 1} + \frac{\rho u^2}{2},$$

(32)

where $\gamma = 1.4$ is the adiabatic index and $p$ is the pressure.

To regularize Eq. (31), we add a viscous term $\left(\nu(\rho, \rho u, E)\right)_x$, where the coefficient $\nu$ is obtained using the entropy viscosity method. Thus, the viscous Euler’s equations can be written as

$$\left(\begin{array}{c} \rho \\ \rho u \\ E \end{array}\right)_t + \left(\begin{array}{c} \rho u - \nu \rho_x \\ \rho u^2 + p - \nu(\rho u)_x \\ (E + p)u - \nu E_x \end{array}\right)_x = \left(\begin{array}{c} 0 \\ 0 \\ 0 \end{array}\right).$$

(33)

We note that an alternative to this simple viscosity would be to use the full Navier–Stokes equations.

5.3.1 Entropy Viscosity (EV) Method for 1D Euler’s Equations

The discretized viscosity coefficient $\nu = \nu_h$ is given in terms of primitive variables $\rho, p,$ and $u$,

$$\nu_h = \min(\nu_{max}, \nu_{EV}),$$

(34)

$$\nu_{EV} = \alpha_{EV} h_x \rho_h(x, t)|r_{EV}(x, t)|,$$

(35)

$$\nu_{max} = \alpha_{max} h_x \rho_h(x, t) \max_{y \in D} \left(|u_h(y, t)| + \sqrt{\gamma T_h(y, t)}\right),$$

(36)

where $T_h = p_h/\rho_h$ is the temperature, $h_x$ is the grid size and

$$r_{EV} = \partial_t S_h + \partial_x (uS)_h,$$

(37)

is the entropy residual for the entropy function $S_h(p_h, \rho_h) = \frac{\rho_h}{\gamma - 1} \log \left(\frac{\rho_h}{\rho_t}\right)$ and its corresponding entropy flux $(uS)_h$.

5.3.2 An Improved Entropy Viscosity

The entropy viscosity method discretizes the entropy residual using the numerical solution. In theory, the entropy residual is large at shocks, and zero at contact discontinuities (where no artificial viscosity is needed). However, our experience is that the discretization of the entropy equation may also trigger the maximum viscosity at contact discontinuities. To the left in Fig. 5, we see a space-time diagram of the entropy residual for Sod’s problem in logarithm scale. Note that a relatively large amount of residual is produced at the contact discontinuity.

To eliminate this undesired behavior along the contact discontinuity, we use the fact that the velocity of the fluid, $u$, is a Riemann invariant along the second characteristic field. Since $u$ is continuous at the contact discontinuity but not at a shock, the product of the increment
Fig. 5  Space time diagram of the magnitude of entropy residual $|r_E|$ (left) and $|\Delta u r_E|$ (right) for Sod’s problem. Blue is small, red is big. Simulations performed with $\nu_S \propto r_E$ (left) and $\nu_S \propto \Delta u r_E$ (right). With the new sensor $|\Delta u r_E|$, the residual, hence the viscosity is driven to zero along contact discontinuity (thicker red line in the middle disappears with the new sensor) (Color figure online)

Fig. 6  Numerical solution (dotted lines) obtained using entropy viscosity proportional to $|r_E|$ (left) and $|\Delta u r_E|$ (right) for Sod’s problem are plotted against the exact solution (solid lines). We plot density (blue), velocity (green), and pressure (red). Contact discontinuity is sharper using the new sensor $|\Delta u r_E|$ (Color figure online)

in the velocity $\Delta u_j = u_{j+1} - u_j$ and $(r_E)_j$ is small at contact discontinuities but still large at shocks. We incorporate the term $\Delta u$ into the improved entropy viscosity

$$v_{EV} = \alpha_{EV} \, h_x \, \rho_h(x, t) |\Delta u||r_{EV}(x, t)|,$$

resulting in a sharper gradient across contact discontinuity, see Fig. 6.

To this end we take $v_E$ and $v_{max}$ as a piecewise constant function on each cell. Thus, we compute the discretized density $\rho_h$, velocity $u_h$, temperature $T_h$, entropy function $S_h$ and entropy flux $u S_h$ at cell center in pointwise manner. Now, to get the entropy residual given in (37), we compute temporal and spatial derivatives using finite differences. Using the notation $S_h = S^n_j$ to denote the approximate flux function $S$ at $x = x_j$, $t = t_n$, we discretize the term $\partial_t S^n_j$ using second order backward difference formula

$$\partial_t S^n_j = \frac{3 S^n_j - 4 S^{n-1}_j + S^{n-2}_j}{2 \Delta t}.$$

Similarly, the term $\partial_x (u S)^n_j$ is approximated by the centered finite difference

$$\partial_x (u S)^n_j = \frac{(u S)^n_{j+1} - (u S)^n_{j-1}}{2 h_x}.$$
Table 5  Convergence study on Euler’s equation with stationary shock

| $h_x$ | 1.25(−02) | 6.25(−03) | 3.13(−03) | 1.56(−03) | 7.81(−04) | 3.91(−04) |
|-------|------------|------------|------------|------------|------------|------------|
| $L_1$ error ($m = 3$) | 1.47(−03) | 9.98(−04) | 7.19(−04) | 5.38(−04) | 3.45(−04) | 2.12(−04) |
| Rate ($m = 3$) | 0.56 | 0.47 | 0.42 | 0.64 | 0.70 | |
| $L_1$ error ($m = 4$) | 1.43(−03) | 1.11(−03) | 8.09(−04) | 5.76(−04) | 3.73(−04) | 2.35(−04) |
| Rate ($m = 4$) | 0.36 | 0.46 | 0.49 | 0.63 | 0.66 | |

5.4 Experiments in One Dimension with Euler’s Equations

We now present results obtained using our Hermite-RK4-EV method for a stationary shock, the Lax, the Sod, and the Shu–Osher problem. For experiments where we use more than one resolution, the EV parameters are tuned on the coarsest grid. In the 1D Euler’s equations simulations, the timestep is chosen as $\Delta t = \text{CFL} \frac{h_x}{\max x |(u \pm c)(x, 0)|}$, where $c = \sqrt{\gamma p/\rho}$ is the speed of sound, with CFL values given in Table 5.

5.4.1 Stationary Shock Problem

By solving the Riemann problem, we decide the states corresponding to a stationary shock. The goal of this experiment is to investigate the stability and accuracy of EV in the presence of shocks. Since small oscillations coming from shocks could potentially pollute the “smooth” regions, this is also a test for how well EV removes numerical artifacts. The computational domain is $D = [-0.5, 0.5]$ with the stationary shock given by

$$(\rho, u, p)(x, t) = \begin{cases} (0.84, 1.08, 0.56) & x < 0, \\ (1, 0.9, 0.71) & x > 0. \end{cases}$$

(39)

The boundary condition are imposed by setting the solution at the boundary so that it coincides with the solution at initial time. We perform a grid refinement study and report the errors in the density in Table 5. We also present the ratio between successive errors. It is unclear why the error corresponding to the $m = 3$ Hermite method is smaller than the error for the $m = 4$ (higher order) method. One possible explanation for this behavior is that the EV parameters must be adjusted for each method.

5.4.2 Lax’s and Sod’s Shock Tube Problems

Lax’s and Sod’s problems come from physical experiments in which a gas tube is separated by a membrane into two sections. The gas in each section is uniform in the $y$ and $z$ direction, so the problem is modeled as a 1-dimensional shock tube. The gas in the left section is kept at a different state than the gas in the right section. At time $t = 0$, the membrane is punctured. In the problem setup, the Euler’s equations are solved on the domain $D = [-0.5, 0.5]$ with initial data

$$(\rho, u, p)(x, 0) = \begin{cases} (0.445, 0.698, 3.528) & x < 0 \\ (0.5, 0, 0.571) & x > 0 \end{cases}$$

(40)

for Lax, and
Fig. 7 Left: lax shock tube, right: sod shock tube. Dashed lines are the numerical solutions, solid lines are the exact solutions. Numerical solutions are obtained using $N_x = 100$ elements. The color blue represents density, green represents velocity, and red represents pressure (Color figure online)

| Problem            | CFL  | $m$ | $\alpha_{EV}$ | $\alpha_{max}$ |
|--------------------|------|-----|---------------|----------------|
| Lax                | 0.2  | 3   | 0.5           | 0.08           |
| Sod                | 0.15 | 3   | 0.2           | 0.08           |
| Shu–Osher          | 0.15 | 3   | 0.01          | 0.05           |
| Shu–Osher          | 0.15 | 4   | 0.03          | 0.03           |
| Stationary shock   | 0.2  | 3, 4| 5             | 0.3            |

\[(\rho, u, p)(x, 0) = \begin{cases} 
(1, 0, 1) & x < 0 \\
(0.125, 0, 0.1) & x > 0 
\end{cases} \tag{41}\]

for Sod. For both problems, we impose fixed boundary condition so that the solution on the boundary is the same as at the initial time. The solution is computed up to time $t = 0.16$ for Lax’s problem and time $t = 0.1644$ for Sod’s problem.

The solution to Riemann problems such as Lax’s and Sod’s shock tubes contains 3 waves propagating from the discontinuity at the initial time. The second wave is a contact discontinuity, where the discontinuity is translated over time. The first and third waves are nonlinear, and can take either rarefaction waves or shock waves.

The results for density $\rho$, velocity $u$ and pressure $p$ are plotted against the exact solution in Fig. 7. In each plot, we use $N_x = 100$ elements. The entropy viscosity parameters used are given in Table 6. In both problems, the shocks are resolved better than the contact discontinuities. Although the shock strength is only of medium size for both problems, some experts considered these tough test cases for non-characteristic-based high order schemes [23].

5.4.3 The Shu–Osher Problem

The Shu–Osher problem poses difficulties for numerical methods due to the sine wave interacting with the shock. Here the domain is $D = [-5, 5]$ with initial data
Fig. 8  Shu–Osher problem. Left: solutions obtained using the \( m = 3 \) Hermite method. Dashed lines are the numerical solutions, solid lines are the “exact” solutions. Numerical solutions are computed using \( N_x = 80 \) elements, “exact” solutions are computed on a much finer grid, with \( N_x = 1280 \) elements. The color blue represents density, green represents velocity, and red represents pressure. Right: comparison of the (zoomed-in) computed density obtained using \( m = 3 \) (solid blue) and \( m = 4 \) (dashed black), both with \( N_x = 1280 \) elements (Color figure online)

\[(\rho, u, p)(x, 0) = \begin{cases} (3.86, 2.63, 10.33) & x < -4, \\ (1 + 0.2 \sin(5x), 0, 1) & x > -4, \end{cases} \tag{42} \]

and with fixed boundary condition so that the solution on the boundary coincides with the solution at the initial time. The solution is computed up to time \( t = 1.8 \) and compared against a computed solution on a much finer grid. We use \( N_x = 80 \) to obtain the numerical solution in Fig. 8, where we interpolate the solution on to a finer grid. The “exact” solution is computed on a grid with \( N_x = 1280 \). Note that even if we use a coarse grid, we can still get roughly the shape of the solution, especially away from the shock. However, when smooth waves are present (see blue oscillatory line to the left of shock) and too close to the shock, these waves get damped.

5.5 Euler’s Equations in Two Dimensions

The two dimensional inviscid Euler equations are given by

\[
\begin{pmatrix}
\rho \\
\rho u \\
\rho v \\
E
\end{pmatrix}_t + \begin{pmatrix}
\rho u \\
\rho u^2 + p \\
\rho u v \\
(E + p)u
\end{pmatrix}_x + \begin{pmatrix}
\rho v \\
\rho u v \\
\rho v^2 + p \\
(E + pv)v
\end{pmatrix}_y = \begin{pmatrix}
0 \\
0 \\
0 \\
0
\end{pmatrix}. \tag{43}
\]

Here, \( \rho \) is the mass density, \( \rho u \) and \( \rho v \) are the momentum density, \( u \) and \( v \) are the velocity in \( x \) and \( y \) directions respectively and \( E \) is the internal energy per unit mass. Furthermore, we assume an ideal gas with equation of state

\[
E = \frac{p}{\gamma - 1} + \frac{\rho (u^2 + v^2)}{2}, \tag{44}
\]

where \( \gamma = 1.4 \) is the adiabatic index and \( p \) is the pressure. For all experiments below, the timestep is chosen as \( \Delta t = CFL \cdot h_x / \max_x |(u \pm c)(x, 0)| \), where \( c = \sqrt{\gamma p / \rho} \) is the speed of sound, with CFL values given in Table 7. We then add viscosity term \( \nabla \cdot (\nu \nabla (\rho, \rho u, \rho v, E)^T) \) to regularize the inviscid equation.
Table 7 Parameters for examples in 2D Euler’s equations

| Problem             | CFL | \( m \) | \( \alpha_{EV} \) | \( \alpha_{max} \) |
|---------------------|-----|---------|-------------------|-------------------|
| Explosion/implosion | 0.2 | 3       | 0.1               | 0.2               |
| Vortex–shock interaction 1 | 0.2 | 3       | 0.01              | 0.04              |
| Vortex–shock interaction 2 | 0.2 | 3       | 0.05              | 0.07              |
| Jet                 | 0.2 | 3       | 0.03              | 0.2               |

5.6 Entropy Viscosity Method for Euler’s Equations in Two Dimensions

The entropy viscosity is identical to the 1D version given in (34), with the exception that it takes the velocity in both directions into account.

\[
\nu_h = \min (\nu_{max}, \nu_{EV}), \tag{45}
\]

\[
\nu_{EV} = \alpha_{EV} h \rho_h (x, t) (|\Delta u| + |\Delta v|) r_{EV} (x, t), \tag{46}
\]

\[
\nu_{max} = \alpha_{max} h \rho_h (x, t) \max_{y \in D} \left( \sqrt{u_h^2(y, t) + v_h^2(y, t) + \gamma T_h(y, t)} \right), \tag{47}
\]

where \( T_h = p_h/\rho_h \) is the temperature, \( h = \min(h_x, h_y) \) is the grid size, \( \Delta u \) and \( \Delta v \) are the value of the jumps in the velocity in \( x \) and \( y \) directions respectively, and

\[
r_{EV} = \partial_t S_h + ((u S)_h)_x + ((v S)_h)_y \geq 0. \tag{48}
\]

To discretize the entropy residual \( r_{EV} \), we again use BDF for the time derivative and centered finite differences for the spatial derivatives,

\[
\partial_x (u S)^n_{jk} = \frac{(u S)^{n}_{j+1,k} - (u S)^{n}_{j-1,k}}{2h_x}, \tag{49}
\]

\[
\partial_y (v S)^n_{jk} = \frac{(v S)^{n}_{j,k+1} - (v S)^{n}_{j,k-1}}{2h_y}. \tag{50}
\]

On the domain \([x_L, x_R] \times [y_B, y_T]\), we use the subscript \( jk \) to indicate that the variable attached is evaluated at \( x = x_L + jh_x \) and \( y = y_B + kh_y \).

5.6.1 Explosion/Implosion Test

First we solve a radially symmetric Riemann problem from Toro [24]. The computational domain is \( D = [-1, 1] \times [-1, 1] \), and the initial data corresponding to an expanding wave is

\[
(\rho, u, v, p)(r, t) = \begin{cases} 
(1, 0, 0, 1) & r < 0.4, \\
(1, 0, 0, 0.1) & r > 0.4.
\end{cases} \tag{51}
\]

For an imploding wave, the initial data is,

\[
(\rho, u, v, p)(r, t) = \begin{cases} 
(1, 0, 0, 1) & r > 0.4, \\
(1, 0, 0, 0.1) & r < 0.4.
\end{cases} \tag{52}
\]

The boundary conditions are imposed by setting the solution on the boundary so that it stays the same as the solution at the initial time. The simulation is performed until time \( t = 0.25 \), before any waves reach the boundary of the domain. We plot the 2D solution in Fig. 9.
Fig. 9 Solution to explosion/implosion problem at time $t = 0.25$. To the left: explosion, to the right: implosion. The numerical solutions (circles) are computed using $N_x = N_y = 100$ elements.

Fig. 10 Cross section of the density for radially symmetric problem along $x$ axis at time $t = 0.25$. To the left: explosion, to the right: implosion. The numerical solutions (circles) are computed using $N_x = N_y = 100$ elements, “exact” solutions (solid lines) are computed using $N_x = N_y = 400$ elements.

Fig. 10, we present a cross section of the density at time $t = 0.25$ with $N_x = N_y = 100$ elements against computed “exact” solution obtained with $N_x = N_y = 400$ elements.

5.6.2 Shock Vortex Interaction

Next we consider the interaction of a shock and a vortex. In general shock–vortex interactions can produce small scales in the form of acoustic waves, and other interesting wave patterns. It has received substantial interest in the literature, see for example [5,6,21,26].

In this experiment, a strong stationary shock with Mach number $2/\sqrt{1.4} \approx 1.69$ collides with a weak vortex with a Mach number $6/2\pi \approx 0.81$. The computational domain is $D = [-9, 3] \times [-4, 4]$ and the initial data is

$$\begin{align*}
(r, u, v, p)(r, t) &= \begin{cases} 
(\rho_{vor}, u_{vor}, v_{vor}, p_{vor}) & x > -4, \\
(2.18, -0.92, 0, 3.17) & x < -4,
\end{cases} 
\end{align*}$$

(53)

where

$$\begin{align*}
\rho_{vor} &= \left[ 1 - \frac{(\gamma - 1)\beta^2}{8\gamma\pi^2} e^{1-x^2-y^2} \right]^{1/(\gamma-1)} 
\end{align*}$$

(54)

$$u_{vor} = 2 - \frac{\beta}{2\pi} ye^{(1-x^2-y^2)/2}$$

(55)
The density schlieren at different times, from top to bottom $t \approx 1.97, t \approx 2.95, t \approx 3.94$ and $t \approx 4.92$. Left: vortex shock interaction 1, right: vortex interaction 2, with parameters given in Table 7. The numerical solutions are obtained with $N_x = 720, N_y = 480$ elements.
\[ v_{\text{vor}} = \frac{\beta}{2\pi} xe^{(1-x^2-y^2)/2} \]
\[ p_{\text{vor}} = \rho^\gamma. \]

and \( \beta = 6. \)

As the vortex passes through the shock, the shock is distorted and the vortex is compressed into an elliptical shape. This phenomena is due to the fact that the vortex is relatively weak compared to the shock. The results are consistent with [26]. In Fig. 11, we compare snapshots of the density Schlieren using two different sets of entropy viscosity parameters, see Table 7. Although the schlierens are plotted on the same color scale, notice that the structures are more pronounced in the pictures on the right column.

5.6.3 Fluid Flow in Jet

As a final example we simulate a planar Mach \( 2/\sqrt{1.4} \approx 1.69 \) jet. The domain \( D = [-15, 55] \times [-17.5, 17.5] \) is discretized using of \( 500 \times 250 \) cells. The initial data is

\[ (\rho, u, v, p)(x, y, t) = (1, 0, 0, 1). \] (58)

We model the jet nozzle by a simple momentum forcing over a \( 1 \times 1 \) patch at the left edge of the computational domain. The jet is started impulsively causing a relatively strong compression wave to be generated. This wave sharpens up to a shock wave that is handled by the entropy viscosity as it is propagated from the nozzle and out into a damping absorbing layer of super-grid type, see [2].

In Figs. 12, 13 and 14 we display snapshots of the vorticity, dilatation and density Schlieren. We note that the viscosity we use here is purely for the regularization of shocks so there is no reason to believe that the flow that we compute resembles reality. Nevertheless, the example illustrates the new methods ability to handle rapidly started flows. Also, it is likely that the particular form of the artificial viscosity does not effect the robustness of the method.

Fig. 12 The vorticity at different times, from left to right, top to bottom \( t = 103.26, t = 118.93, t = 134.89, \) and \( t = 150.14. \) The numerical solutions are obtained with \( N_x = 500, N_y = 250 \) elements.
Fig. 13 The dilatation at different times, from left to right, top to bottom $t = 103.26$, $t = 118.93$, $t = 134.89$, and $t = 150.14$. The numerical solutions are obtained with $N_x = 500$, $N_y = 250$ elements.

Fig. 14 The density schlieren at different times, from left to right, top to bottom $t = 103.26$, $t = 118.93$, $t = 134.89$, and $t = 150.14$. The numerical solutions are obtained with $N_x = 500$, $N_y = 250$ elements.

6 Conclusion

In conclusion we have demonstrated that flux-conservative Hermite methods are suitable for solving nonlinear conservation laws, especially in the presence of shocks. The new methods still converge at a rate of $(2m + 1)$ for smooth problems.

The adaptation of the entropy viscosity method to Hermite methods successfully suppresses oscillations near shocks, but we find that our current implementation is quite dissipative when solving the Shu–Osher problem. For contact waves we proposed a modification to the entropy viscosity method which eliminates a large amount of the spurious viscosity at contact discontinuities.
References

1. Adeline, K., Daniel, A.: On the scaling of entropy viscosity in high order methods. In: Springer Lecture Notes in Computational Science and Engineering (in press).
2. Appelö, D., Colonius, T.: A high-order super-grid-scale absorbing layer and its application to linear hyperbolic systems. J. Comput. Phys. 228(11), 4200–4217 (2009).
3. Appelö, D., Hagstrom, T.: CHIDES: the Charles Hermite interpolation differential equation solver. http://www.chides.org.
4. Brooks, A.N., Hughes, T.J.R.: Streamline upwind/Petrov–Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations. Comput. Methods Appl. Mech. Eng. 32(1), 199–259 (1982).
5. Denet, B., Biamino, L., Lodato, G., Vervisch, L., Clavin, P.: Model equation for the dynamics of wrinkled shockwaves: comparison with DNS and experiments. Combust. Sci. Technol. 187(1–2), 296–323 (2015).
6. Dumbser, M., Zanotti, O., Loubère, R., Diot, S.: A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws. J. Comput. Phys. 278, 47–75 (2014).
7. Goodrich, J., Hagstrom, T., Lorenz, J.: Hermite methods for hyperbolic initial-boundary value problems. Math Comput 75(254), 595–630 (2006).
8. Guermond, J.-L., Pasquetti, R.: Entropy viscosity method for higher-order approximations of conservation laws. Lect. Notes Comput. Sci. Eng. 76, 411–418 (2011).
9. Guermond, J.-L., Pasquetti, R.: Entropy-based nonlinear viscosity for Fourier approximations of conservation laws. C. R. Math. 346(13), 801–806 (2008).
10. Hagstrom, T., Appelö, D.: Experiments with Hermite methods for simulating compressible flows: Runge–Kutta time-stepping and absorbing layers. In: 13th AIAA/CEAS Aeroacoustics Conference, number 2007-3507 (2007).
11. Hagstrom, T., Appelö, D.: Solving PDEs with Hermite interpolation. In: Springer Lecture Notes in Computational Science and Engineering (2015).
12. Jiang, G.-S., Shu, C.-W.: Efficient implementation of weighted ENO schemes. J. Comput. Phys. 126(1), 202–228 (1996).
13. Johnson, C., Szepessy, A., Hansbo, P.: On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws. Math. Comput. 54(189), 107–129 (1990).
14. Klockner, A., Warburton, T., Hesthaven, J.S.: Viscous shock capturing in a time-explicit discontinuous Galerkin method. Math. Model. Nat. Phenom. 6(3), 57–83 (2011).
15. Krivodonova, L.: Limiters for high-order discontinuous Galerkin methods. J. Comput. Phys. 226(1), 879–896 (2007).
16. LeVeque, R.J.: Numerical Methods for Conservation Laws, 2nd edn. Birkhäuser Verlag, Basel (1992).
17. LeVeque, R.J.: Finite Volume Methods for Hyperbolic Problems. Cambridge University Press, Cambridge (2002).
18. Liu, X.-D., Osher, S., Chan, T.: Weighted essentially non-oscillatory schemes. J. Comput. Phys. 115(1), 200–212 (1994).
19. Martín, M.P., Taylor, E.M., Wu, M., Weirs, V.G.: A bandwidth-optimized WENO scheme for the effective direct numerical simulation of compressible turbulence. J. Comput. Phys. 220(1), 270–289 (2006).
20. Persson, P.O., Peraire, J.: Sub-cell shock capturing for discontinuous Galerkin methods. In: 44 th AIAA Aerospace Sciences Meeting and Exhibit, pp. 1–13 (2006).
21. Rault, A., Chiavassa, G., Donat, R.: Shock–vortex interactions at high Mach numbers. J. Sci. Comput. 19(1–3), 347–371 (2003).
22. Shu, C.-W.: Essentially Non-oscillatory and Weighted Essentially Non-oscillatory Schemes for Hyperbolic Conservation Laws. Springer, Berlin (1998).
23. Shu, C.-W., Osher, S.: Efficient implementation of essentially non-oscillatory shock-capturing schemes. J. Comput. Phys. 77(2), 439–471 (1988).
24. Toro, E.F.: Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction. Springer Science & Business Media, Berlin (2013).
25. Von Neumann, J., Richtmyer, R.D.: A method for the numerical calculation of hydrodynamic shocks. J. Appl. Phys. 21, 232–237 (1950).
26. Zhang, S., Zhang, Y.-T., Shu, C.-W.: Multistage interaction of a shock wave and a strong vortex. Phys. Fluids (1994-present), 17(11), 116101 (2005).
27. Zingan, V., Guermond, J.-L., Morel, J., Popov, B.: Implementation of the entropy viscosity method with the discontinuous Galerkin method. Comput. Methods Appl. Mech. Eng. 253, 479–490 (2013).