Solving PDEs with Hermite Interpolation

Thomas Hagstrom and Daniel Appelö

Abstract We examine the use of Hermite interpolation, that is interpolation using derivative data, in place of Lagrange interpolation to develop high-order PDE solvers. The fundamental properties of Hermite interpolation are recalled, with an emphasis on their smoothing effect and robust performance for nonsmooth functions. Examples from the CHIDES library are presented to illustrate the construction and performance of Hermite methods for basic wave propagation problems.

1 Introduction

Polynomials are the workhorse for approximating the solution to general PDE's - indeed, using Taylor expansions, it is clear that convergence of a method at high order with grid refinement is equivalent to it being at least approximately exact for polynomial solutions of high degree. Thus both high order finite difference methods and nodal spectral element methods are typically constructed using Lagrange interpolants. However, the two classes of method are obviously distinct in the way the polynomials are used - for difference methods they are implicitly reconstructed at each grid point via the difference formulas, while for element based approaches they are defined and used in a finite region. An advantage of the element-based interpolants is the possibility to directly use properties of the PDE to guarantee stability, as in the standard continuous and discontinuous Galerkin frameworks [11, 18], as well as the localization of much of the computational effort. A disadvantage, however, is the fact that high-degree polynomials can support boundary layers at element edges, as illustrated by the plot of the degree 15 Legendre polynomial and its derivative in Figure 1.

Thomas Hagstrom
Southern Methodist University, Dallas TX USA e-mail: thagstrom@smu.edu

Daniel Appelö
The University of New Mexico, Albuquerque NM USA e-mail: appelo@unm.edu

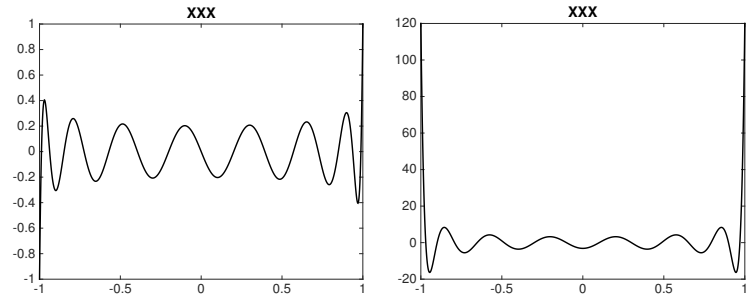

Fig. 1 Plot of the Legendre polynomial, $P_{15}(x)$ and its first derivative. Note that the maximum value is 1 and the maximum derivative value is 120 - see (1).

This boundary layer phenomenon is encapsulated in the inequalities of Bernstein and Markov [5]:

Theorem 1. *Let $q(x)$ be a polynomial of degree n. Then for $-1 \leq x \leq 1$*

$$\left| \frac{dq}{dx} \right| \leq \min\left(\frac{n}{\sqrt{1-x^2}}, n^2 \right) \|q\|_{L^\infty([-1,1])}. \tag{1}$$

The practical consequence of (1) and its generalization to multidimensional elements [14] is that differentiation matrices built from polynomials of degree n must have first derivative matrices whose norm scales like $\frac{n^2}{H}$, where H is the element width. Given that the element contains $n+1$ Lagrange nodes, this is a factor of n worse than the scaling of finite difference formulas with a comparable node density, leading to an artificially stiff semidiscretization. If second derivatives are present the situation becomes more extreme. Although approaches based on mappings (to produce nonpolynomial bases, e.g. [21]) or filtering [26] can be used, the fundamental fact is that a nonstiff polynomial differentiation matrix can only be based on differentiation near the element center.

Motivated by these facts, and in addition by the inherent stability properties detailed below, we propose the use of **Hermite interpolation** in place of Lagrange interpolation to construct high-order polynomial elements. That is, rather than using function values distributed throughout an element as the basic degrees-of-freedom, we use function and derivative values, or equivalently the coefficients of the Taylor polynomial, centered at an interior point. In many aspects the resulting methods enjoy the advantages of both finite difference and finite element discretizations:

i. Degree-independent stability constraints - with sufficiently accurate local time-stepping for hyperbolic problems all degrees-of-freedom in an element can be updated independent of neighboring elements over a time step limited only by domain-of-dependence requirements.
ii. Stability based on continuous energy estimates.
iii. Highly localized evolution of many degrees-of-freedom.

Below, through simple examples, we will illustrate the basics of a PDE solver built on Hermite interpolation. The examples are implemented as Matlab programs and freely available as part of the CHIDES[1] library `chides.org`. Our initial release of CHIDES will contain, besides the Matlab implementation of the examples discussed here, various subroutines and drivers written in modern FORTRAN illustrating and enabling the construction of Hermite PDE solvers on structured meshes. We plan future releases including more complex capabilities such as coupling with DG methods on hybrid grids, as well as implementations on overset grids.

2 Hermite Interpolation

Theorem 2 (Hermite interpolation (Dahlquist & Björk [12])). *Let $\{x_i\}_{i=1}^s$ be s distinct points. Let $f(x)$ be a function defined and with derivatives up to order m_i at x_i. Then there exists a unique polynomial $p(x)$ of degree $\leq r-1$, where $r = \sum_{i=1}^s (m_i+1)$ solving the **Hermite interpolation problem**:*

$$\left.\frac{d^j p(x)}{dx_j}\right|_{x=x_i} = \left.\frac{d^j f(x)}{dx_j}\right|_{x=x_i}, \quad j=0,\ldots,m_i, \;\; i=1,\ldots,s. \tag{2}$$

Piecewise interpolation
Now consider the special form of Hermite interpolation used in CHIDES - namely piecewise interpolation using two nodes with m derivatives at each. Suppose $x_0 < x_1 < \ldots < x_N$. On an interval (x_{i-1}, x_i) we independently compute an interpolant, $p_i(x)$, of degree $2m+1$, satisfying (2) with $m_{i-1} = m_i = m$. The global piecewise interpolant we denote by:

$$\mathscr{I}_m f = p_i(x), \;\; x \in (x_{i-1}, x_i). \tag{3}$$

Note that $\mathscr{I}_m f \in C^m$. We also employ piecewise degree $2m+1$ interpolation on a dual grid consisting of nodes $x_{i+1/2} = (x_i + x_{i+1})/2$ and define

$$\tilde{\mathscr{I}}_m f = p_{i+1/2}(x), \;\; x \in (x_{i-1/2}, x_{i+1/2}) \tag{4}$$

with $p_{i+1/2}(x)$ being the solution to the Hermite interpolation problem with data consisting of derivatives through order m at $x_{i\pm 1/2}$.

Newton form
The cellwise Hermite interpolation problem is solved repeatedly during each time step, and its cost is the dominant cost for linear systems with constant coefficients. An efficient way to solve to (2) is to form the generalized divided difference table used to find the interpolating Newton polynomial. We form the Newton table by first filling in $f^{(s)}(x_{i-1})/s!$ and $f^{(s)}(x_i)/s!$, $s = 0,\ldots,m$ as illustrated in Table 1.

[1] **C**harles **H**ermite **I**nterpolation **D**ifferential **E**quation **S**olver

Next we fill in the missing positions (indicated by $\star$ in Table 1) one column at a time from left to right. The interpolating polynomial, $p_i(x)$, can then be found as

$$p_i(x) = a_0 + a_1(x-x_{i-1}) + \cdots + a_{m+1}(x-x_{i-1})^{m+1} + a_{m+2}(x-x_{i-1})^{m+1}(x-x_i) \\ + \cdots + a_{2m+1}(x-x_{i-1})^{m+1}(x-x_i)^m,$$

where $a_j, j = 0,\ldots,2m+1$ are the coefficients on the upper diagonal in the table.

Table 1 A generalized Newton divided difference table.

$$\begin{array}{cccccccc} x_{i-1} & f(x_{i-1}) & f^{(1)}(x_{i-1})/1! & f^{(2)}(x_{i-1})/2! & \star & \star & & \\ x_{i-1} & f(x_{i-1}) & f^{(1)}(x_{i-1})/1! & \star & \star & \star & \star & \\ x_{i-1} & f(x_{i-1}) & \star & \star & \star & \star & \star & \\ x_i & f(x_i) & \star & \star & \star & & & \\ x_i & f(x_i) & f^{(1)}(x_i)/1! & f^{(2)}(x_i)/2! & & & & \\ x_i & f(x_i) & f^{(1)}(x_i)/1! & & & & & \end{array}$$

In our PDE solvers we work with monomial basis,

$$p_i(x) = \sum_{j=0}^{2m+1} c_j x^j. \tag{5}$$

The coefficients c_j can be obtained from a_j by a fast dual Vandermonde solve [12].

Error estimates

Detailed formulas for the error in Hermite interpolation of a smooth function are given in [4]. Precisely, for $x \in (x_{i-1}, x_i)$, the Peano representation of the local error can be easily derived by noting that $e = f - \mathscr{I}_m f$ solves the two point boundary value problem

$$\frac{d^{2m+2}e}{dx^{2m+2}} = \frac{d^{2m+2}f}{dx^{2m+2}}, \quad \frac{d^j e}{dx^j} = 0, \quad x = x_{i-1}, x_i, \quad j = 0,\ldots m. \tag{6}$$

Thus

$$f(x) - \mathscr{I}_m f(x) = \int_{x_{i-1}}^{x_i} K_i(x,s) \frac{d^{2m+2}f}{dx^{2m+2}}(s) ds, \tag{7}$$

where the kernel K_i is the Green's function for the two-point boundary value problem (6). Indeed, the local Hermite interpolant can be characterized as the unique solution of the inhomogeneous boundary value problem

$$\frac{d^{2m+2}p_i}{dx^{2m+2}} = 0, \quad \frac{d^j p_i}{dx^j} = \frac{d^j f}{dx^j}, \quad x = x_{i-1}, x_i, \quad j = 0,\ldots m. \tag{8}$$

Simple scaling arguments combined with the transformation $x = x_{i-1} + z h_i$ then show that $e = O(h_i^{2m+2})$ where $h_i = x_i - x_{i-1}$ is the element width. We also have the formula

$$f(x) - \mathscr{I}_m f(x) = \frac{(-1)^{m+1}}{(2m+2)!}(x-x_{i-1})^{m+1}(x_i-x)^{m+1}\frac{d^{2m+2}f}{dx^{2m+2}}(\eta). \tag{9}$$

These formulas show that the error is significantly smaller near the endpoints of the interval, and allows one to compute an accurate artificial dissipation coefficient in a modified equation approximation to the discrete evolution; see [3, 20] for details.

Smoothing properties

A fundamental feature of piecewise Hermite interpolation is the following minimization property in the H^{m+1} seminorm,

$$|w|_{m+1}^2 \equiv \int_{x_0}^{x_N} \left(\frac{d^{m+1}w}{dx^{m+1}}\right)^2 dx. \tag{10}$$

Theorem 3. *Suppose g is any function in $H^{m+1}(x_0,x_N)$ satisfying $\frac{d^j g}{dx^j}(x_i) = \frac{d^j f}{dx^j}(x_i)$, $j = 0,\ldots,m$, $i = 0,\ldots,N$. Then $|\mathscr{I}_m f|_{m+1} \le |g|_{m+1}$.*

This result holds locally on each interval and follows from the fact that $p_i(x)$ is **orthogonal** in the H^{m+1} semi-inner product to any function $w(x)$ satisfying $\frac{d^j w}{dx^j}(x_{i-1}) = \frac{d^j w}{dx^j}(x_i) = 0$, $j = 0,\ldots,m$. In fact by the Pythagorean Theorem

$$|f|_{m+1}^2 = |\mathscr{I}_m f|_{m+1}^2 + |f - \mathscr{I}_m f|_{m+1}^2. \tag{11}$$

These smoothing results are used to prove the stability of Hermite methods and establish optimal convergence results; see [16] and the discussion below.

Application to nonsmooth functions

The aforementioned smoothing properties of Hermite interpolation are also beneficial when dealing with nonsmooth functions. For example consider the canonical model of a shock wave, the step function $q(x) = -\text{sign}(x)$. Let $Q(x)$ be the Hermite interpolant of degree $2m+1$ of $q(x)$ on $x \in [-1,1]$. It is straightforward to prove (see [3]) that $Q(x)$ is monotone and thus the total variation of $Q(x)$ is identical to the total variation of $q(x)$. The first 20 Hermite interpolants are displayed in Figure 2. A well-known result due to Bernstein is that the

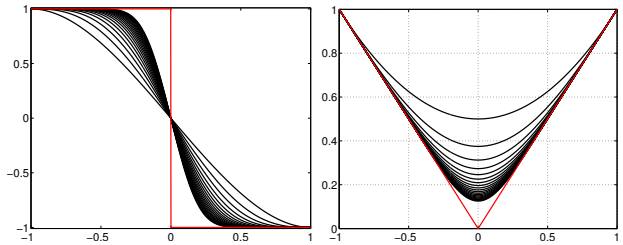

Fig. 2 Hermite interpolating polynomials of degree $2m+1, m = 1,\ldots,20$ of the step function $q(x) = -\text{sign}(x)$ (to the left) and the absolute value function $|x|$ (to the right.)

sequence of Lagrange interpolation polynomials for $|x|$ at equally spaced nodes in $x \in [-1,1]$ diverges everywhere, except at zero and the end-points. As can be seen in Figure 2 Hermite interpolation does considerably better. In fact, one can check that the degree $2m+1$ Hermite interpolant for $|x|$ coincides with the polynomial

$$b(x) = \sum_{k=0}^{m} \binom{2k}{k} \frac{(-1)^{k+1}(x^2-1)^k}{2^{2k}(2k-1)},$$

which, in turn, is identical to the first terms of the generalized binomial expansion

$$(1+t)^{\frac{1}{2}} = \sum_{k=0}^{\infty} \binom{1/2}{k} t^k,$$

when we set $t = x^2 - 1$ (note that $|x| = (x^2)^{\frac{1}{2}}$). The sequence of Hermite interpolation polynomials thus do converge in $|x| < 1$.

3 A Hermite-Taylor method for solving $u_t + u_x = 0$

We now describe how the approximate solution of a PDE can be found using Hermite interpolation combined with Taylor series approximation in time. The algorithms are implemented in the Matlab files `Hermite_Taylor_1Ddriver.m`, `Advection1D_PDE.m` and `Advection1D_INIT.m` which can be downloaded from `chides.org`.
Consider the scalar advection equation with periodic boundary conditions:

$$\frac{\partial v(x,t)}{\partial t} + c\frac{\partial v(x,t)}{\partial x} = 0, \;\; x \in [x_l, x_r], \;\; t > 0, \tag{12}$$

$$v(x,0) = v_0(x), \;\; v(x_l,t) = v(x_r,t). \tag{13}$$

The first step in our method (implemented in `Hermite_Taylor_1Ddriver.m`) is to define the primal and dual grids with n_x+1 and n_x grid-points covering the computational domain

$$x_i = x_l + ih_x, \;\; h_x = (x_r - x_l)/n_x, \tag{14}$$

with $i = 0,\ldots,n_x$ for the primal grid and $i = 1/2,\ldots,n_x - 1/2$ for the dual grid. Next, we initialize the degrees-of-freedom used to describe the approximate solution, which are approximations to scaled derivatives of the solution of orders $0,\ldots,m$, or equivalently scaled coefficients of the degree-m Taylor polynomial. At $t = 0$ the piecewise degree-$2m+1$ Hermite interpolant $u(x,0)$ is determined by:

$$c_l = \frac{h_x^l}{l!} \left.\frac{d^l u(x,0)}{dx^l}\right|_{x=x_i} \approx \frac{h_x^l}{l!} \left.\frac{d^l v_0}{dx^l}\right|_{x=x_i}.$$

The data to be evolved is stored as an array of coefficients; `u(l,k,i)` holds the coefficient c_l of the kth field at the grid-point x_i. We obtain these basic degrees-of-freedom directly from the initial data. In the example in `Advection1D_INIT.m` we use $v_0(x) = \sin 20\pi x$ and may compute the coefficients directly, but in general we can find them by solving a local interpolation problem at each grid-point.
To evolve the approximate solution in time, we choose a time step Δt satisfying the CFL condition

$$c\Delta t < h_x. \tag{15}$$

Note that the degree m does not appear in this relation. We now form space-time polynomials centered at a grid-point on the dual grid and at time t_n (initially $t_n = 0$)

$$u^n_{i+\frac{1}{2}}(x,t) = \sum_{l=0}^{2m+1} \sum_{s=0}^{q} d_{ls} \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l \left(\frac{t-t_n}{\Delta t} \right)^s. \tag{16}$$

At time $t = t_n$ this expression reduces to

$$u^n_{i+\frac{1}{2}}(x,t_n) = \sum_{l=0}^{2m+1} d_{l0} \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l, \tag{17}$$

where the coefficients d_{l0} in (17) are determined so that (17) is the Hermite interpolant of the data at the adjacent primal nodes.
To find the remaining coefficients d_{ls} we repeatedly differentiate (12) in space and time:

$$\frac{\partial^{l+s} v}{\partial x^l \partial t^s} = -c \frac{\partial^{l+s} v}{\partial x^{l+1} \partial t^{s-1}}, \tag{18}$$

and insist that our approximation u satisfy (18). In particular, note that at $(x_{i+\frac{1}{2}}, t_n)$ the following relation holds

$$d_{ls} = \frac{h_x^l}{l!} \frac{\Delta t^s}{s!} \left. \frac{\partial^l \partial^s u}{\partial x^l \partial t^s} \right|_{x = x_{i+\frac{1}{2}}, t = t_n}, \tag{19}$$

which together with (18) yields the recursion

$$d_{ls} = -c \frac{l+1}{s} \frac{\Delta t}{h_x} d_{l+1\, s-1}, \;\; l = 0, \ldots, 2m+1, s = 1, \ldots, q = 2m+2. \tag{20}$$

Thus, the coefficients d_{ls} are updated recursively. Once the "time-derivative-coefficients" are known we can simply update the approximation at the dual grid-point at the next half time level by evaluating

$$\frac{\partial^l u^n_{i+\frac{1}{2}}}{\partial x^l}(x, t_n + \Delta t/2), \;\; l = 0, \ldots, m,$$

Table 2 Error data for the evolution of $v_0(x) = \sin 20\pi x$ for different methods and final times.

Final time	m	n_x	# time steps	l_2-error		Final time	m	n_x	# time steps	l_2-error
1	1	2000	2222	1.92(-6)		1000	5	20	22222	3.89(-3)
1	5	21	23	2.04(-6)		1000	15	5	5556	9.87(-8)
1	11	6	7	3.73(-7)		1000	25	4	4444	1.16(-9)

Repeating the procedure at the next half time level and using the periodic boundary conditions completes a full time step.
To demonstrate the method we run `Hermite_Taylor_1Ddriver.m` with the initial data $v_0(x) = \sin 20\pi x$. The computational domain is $x \in [0,1]$; thus there are 10 wavelengths inside the computational domain. We choose two different final times: 1 and 1000. This corresponds to waves traveling 10 and 10000 wavelengths respectively. The results for some different combinations of m, n_x with the ratio $\frac{\Delta t}{h_x} = \frac{9}{10}$ are shown in Table 2. The table clearly demonstrates the benefits of using a **very high order method**; for example using a method of order 51 on a grid with 0.4 grid-points per wavelength to evolve the solution 10000 wavelengths using only 4444 timesteps the error is $1.16 \cdot 10^{-9}$.
As a second example we evolve a square wave using $n_x = 8$ and $m = 5, 15, 25$, yielding l_2-errors: 2.99(-2), 4.66(-3) and 6.13(-4) at the final time 10. The approximation and errors are displayed in Figure 3. As the solution is nonsmooth we cannot expect convergence at the full order of the method; see [3] where a discussion of the expected convergence behavior based on a modified equation is given. Despite this we still see a big improvement when a high order method is used.

Convergence analysis

The analysis of convergence for the Hermite-Taylor method implemented above follows from the smoothing and convergence properties of Hermite interpolation combined with the observation that so long as (15) holds the updated Taylor polynomial at the cell center is in fact the Taylor expansion of the **exact solution** of

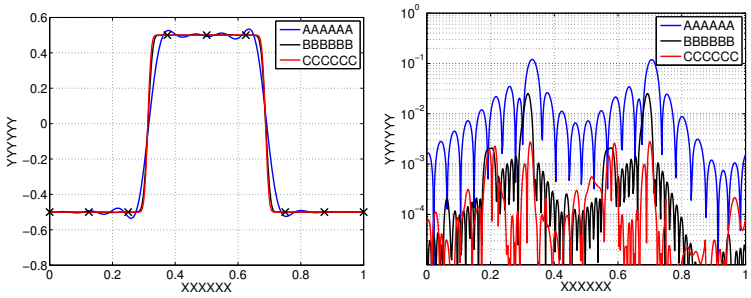

Fig. 3 Left: The square wave `(x>0.25).*(x<0.75)-0.5` at time 10 using $n_x = 8$ and m = 5, 15, 25. The crosses mark the location of the grid points. Right: The error at time 10 as a function of x for the three different choices of m.

the evolution problem over the half time step. Thus we may succinctly express the algorithm as:

$$u_h^{n+1/2} = \tilde{\mathscr{I}}_m S(\Delta t/2) u_h^n, \quad u_h^{n+1} = \mathscr{I}_m S(\Delta t/2) u_h^{n+1/2},$$

where S denotes the exact solution operator for the PDE - in this case simply translation by $c\Delta t/2$. Since S preserves the H^{m+1}-seminorm we immediately conclude from (11) that

$$|u_h^n|_{m+1} \leq |u_h^0|_{m+1}, \tag{21}$$

establishing stability[2]. We can then obtain a slightly suboptimal error estimate in the seminorm by combining (11) and the error bound obtained by taking $m+1$ derivatives of (7). Let $u(x,t)$ represent the true solution and $e^n = u - u_h^n$, $e^{n+1/2} = u - u_h^{n+1/2}$ represent the errors. Then

$$\begin{aligned} e^{n+1/2} &= S(\Delta t/2)u(\cdot,t_n) - \tilde{\mathscr{I}}_m S(\Delta t/2) u_h^n \qquad (22)\\ &= \tilde{\mathscr{I}}_m S(\Delta t/2) e^n + u(\cdot,t_{n+1/2}) - \tilde{\mathscr{I}}_m u(\cdot,t_{n+1/2}) \\ e^{n+1} &= S(\Delta t/2)u(\cdot,t_{n+1/2}) - \mathscr{I}_m S(\Delta t/2) u_h^{n+1/2} \qquad (23)\\ &= \mathscr{I}_m S(\Delta t/2) e^{n+1/2} + u(\cdot,t_{n+1}) - \mathscr{I}_m u(\cdot,t_{n+1}), \end{aligned}$$

which implies

$$|e^{n+1/2}|_{m+1}^2 \leq |e^n|_{m+1}^2 + O(h_x^{2m+2}), \quad |e^{n+1}|_{m+1}^2 \leq |e^{n+1/2}|_{m+1}^2 + O(h_x^{2m+2}).$$

Tracking these inequalities shows that $|e^n|_{m+1} = O(h_x^{m+1/2})$. In fact this argument can be refined to prove the optimal error estimate [16]:

Theorem 4. *There exists a constant, $C(T)$, independent of h_x and the initial data $u(x,0)$ such that for all $n \leq \frac{T}{\Delta t}$*

$$\|e^n\|_{L^2} \leq C h_x^{2m+1} \|u(\cdot,0)\|_{2m+2}. \tag{24}$$

It is also shown in [16] that the result holds in general for constant coefficient symmetric hyperbolic systems in any number of space dimensions. It can also be generalized to variable coefficients and inexact time stepping so long as the local time stepping schemes are sufficiently accurate.

4 Incorporating nonlinearity

For nonlinear PDEs it is often more efficient to use a one-step ODE solver than the Taylor series approach used above. In particular, using Taylor series requires the

[2] We must also use the fact that the average value of the solution remains constant.

repeated differentiation in time of the PDE, spawning many new terms. In contrast a standard ODE solver just requires the computation of a single time derivative. Assume we have found the Hermite interpolant at a dual grid-point $x_{i+\frac{1}{2}}$ but rather than expanding in time let the coefficients d_l be time dependent functions

$$u^n_{i+\frac{1}{2}}(x,t) = \sum_{l=0}^{2m+1} d_l(t) \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l. \tag{25}$$

For a PDE $v_t = f(v)$ we can insert (25):

$$\frac{\partial u^n_{i+\frac{1}{2}}(x,t)}{\partial t} = \sum_{l=0}^{2m+1} d_l'(t) \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l = f(u^n_{i+\frac{1}{2}}(x,t)). \tag{26}$$

As before we can differentiate in space and evaluate at $x = x_{i+\frac{1}{2}}$ to find

$$\frac{k!}{h_x^k} d_k'(t) = \frac{\partial^k}{\partial x^k} f(u^n_{i+\frac{1}{2}}(x,t)) \Big|_{x=x_{i+\frac{1}{2}}}. \tag{27}$$

To avoid the differentiation of the right hand side we first approximate $f(u^n_{i+\frac{1}{2}}(x,t))$ by a Taylor polynomial of degree $2m+1$

$$f(u^n_{i+\frac{1}{2}}(x,t)) \approx \sum_{l=0}^{2m+1} b_l(t) \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l, \tag{28}$$

for which differentiation is straightforward. With this approximation and after carrying out the differentiation in (27) we obtain the local system of ODEs

$$d_k'(t) = b_k(t), \;\; k = 0, \ldots, 2m+1, \tag{29}$$

that can be solved to evolve our approximate solution. Of course, this requires us to first find the Taylor coefficients $b_k(t)$.
The precise way to compute $b_k(t)$ depends on the composition of f . For example, for the nonlinearity vv_x encountered, e.g., in Burgers' equation $v_t + vv_x = \varepsilon v_{xx}$, we may first compute the derivative

$$v_x \approx \sum_{l=1}^{2m+1} \frac{l}{h_x} d_l(t) \left(\frac{x - x_{i+\frac{1}{2}}}{h_x} \right)^l, \tag{30}$$

followed by a polynomial multiplication truncated to degree $2m+1$. This example is implemented in `Burgers1D_PDE.m` and discussed in detail below.
For more general non-linearities we can use techniques for finding recursions for Taylor series. Let $f(x), w(x)$ and $u(x)$ have Taylor series around some base point with coefficients F_k, W_k and U_k. Then, for non-linearities which satisfy the

differential equation $f^{'}(x) = w(x)u^{'}(x)$ (w is a function of f, u or both) we can directly compute the coefficients $F_k, k = 1,2,\ldots$ using the formula [24]

$$F_k = W_0 U_k + \frac{1}{k}\sum_{j=0}^{k-1} jU_j W_{k-j}. \tag{31}$$

For example, if $f = \exp(u)$ we have $f^{'}(x) = f(x)u^{'}(x)$ and thus $w = f$. We start the recursion with $F_0 = \exp(U_0)$.
Thus, for general conservation laws in the form $v_t + (f(v))_x$ we may first use (31) to find a truncated Taylor series followed by differentiation (by the formula (30))[3].

A Hermite-Runge-Kutta solver for $v_t + vv_x = \varepsilon v_{xx}$
To make things concrete we now consider the approximate solution to viscous Burgers' equation using the approach outlined above. We evolve the local system of ODEs (29) using the classic fourth order Runge-Kutta method. The driver routine is called `Hermite_RK_1Ddriver.m` and the routines for the PDE and the initial data are `Burgers1D_PDE.m` and `Burgers1D_INIT.m`.
The nonlinearity in the PDE is handled as outlined above, we first differentiate and then perform a polynomial multiplication (in the code this is done using Matlab's built-in polynomial multiplication routine `conv`.)
The driver `Hermite_RK_1Ddriver.m` is nearly the same as the driver for the Hermite-Taylor method. As before the initial data is set up in a separate file, here in `Burgers1D_INIT.m`. As an example we choose the initial data to be

Table 3 Errors at time 0.2 (left) and 0.35 (right) for Burgers equation for different order methods.

nx	7	9	11	13	15	nx	15	35	55	75	95
Error, $m=3$	3.7(-3)	1.2(-3)	4.0(-4)	1.3(-4)	4.6(-5)	$m=3$	6.3(-2)	2.3(-2)	5.6(-3)	1.2(-3)	2.2(-4)
Rate		4.4	5.5	6.6	7.5	Rate		1.2	3.1	5.0	7.0
Error, $m=5$	5.5(-4)	8.2(-5)	2.1(-5)	7.8(-6)	2.9(-6)	$m=5$	9.6(-2)	1.2(-2)	8.5(-4)	2.1(-5)	1.5(-5)
Rate		7.6	6.8	5.9	6.9	Rate		2.4	5.9	12.0	1.4
Error, $m=7$	1.5(-4)	2.1(-5)	3.4(-6)	5.5(-7)	8.4(-8)	$m=7$	7.6(-2)	3.9(-3)	9.6(-5)	1.1(-5)	6.8(-7)
Rate		7.8	9.0	10.8	13.2	Rate		3.5	8.2	7.1	11.6

$v(x,0) = -sin(\pi x)$ on the domain $x \in [-1,1]$ and $\varepsilon = 0.02$. This data develops into a shock-like sharp transition around time 0.3 so we evaluate the error at 0.2, well before the formation time, and at 0.35, just after the shock forms. In order to maintain stability for this nonlinear problem we reduce $\frac{\Delta t}{h_x}$ to 0.1 and take a single Runge-Kutta substep. We vary the resolution using methods of order 7, 11 and 15; see the results in Table 3. The rate of convergence is not quite at the spatial design order, most likely due to the fourth order accurate time stepper.

[3] Conservation can be enforced when we interpolate, but we have not yet experimented with this approach.

Tol	1.0(-2)	1.0(-3)	1.0(-4)	1.0(-5)	1.0(-6)	1.0(-7)	1.0(-8)	1.0(-9)	1.0(-10)	1.0(-11)	1.0(-12)
l_2-error	1.1(-1)	8.2(-3)	1.5(-3)	1.5(-6)	1.7(-7)	1.6(-9)	3.3(-10)	1.8(-12)	4.2(-14)	1.1(-14)	1.0(-14)

Table 4 Actual errors in the computed solutions for various tolerances for the adaptive Hermite-Runge-Kutta method applied to viscous Burgers' equation.

p-adaptivity

For problems with highly localized features it is often useful to employ adaptive methods. Methods based on Hermite interpolation can be enhanced with both p and H adaptivity and, in particular, incorporating p-adaptivity is quite straightforward. Noting that the above descriptions of the methods are local in the sense that we only require m derivatives at two adjacent nodes in order to evolve the solution a half time step, and also noting that Theorem 3 holds locally, we can allow m to vary spatially choosing $m_{\text{loc}} = \min(m_i, m_{i+1})$ when we form the Hermite interpolant at $x_{i+1/2}$. The driver routine `Padapt_Hermite_RK_1Ddriver.m` illustrates how natural it is to incorporate p-adaptivity.

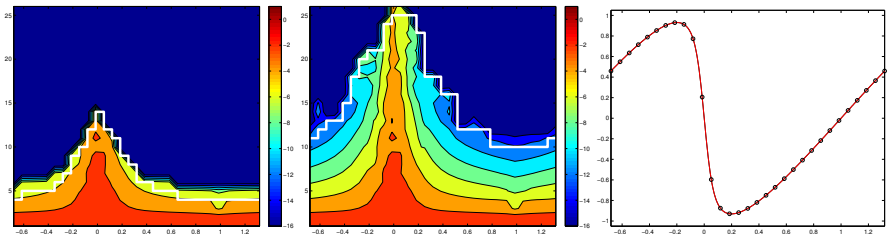

Fig. 4 In the left and middle figures the white line denotes the number of derivatives in the adaptive method that were used at the final time. The contour plot is the base 10 logarithm of the coefficients stored in `u`. The left figure corresponds to `tol = 1e-4` and the middle to `tol = 1e-12`. To the right the computed solution (black line behind the red line), the exact solution (red) and the solution at the 31 nodes (black circles) are displayed.

Taking `mmax = 25` we compute the solution to the Burgers example using various tolerances. As can be seen from the results displayed in Table 4 the algorithm yields solutions with l_2-errors roughly at the level of the selected tolerance.
In Figure 4 we display the solution at the end time 0.35 and the distribution of the number of derivatives used in the computation. Note that in order to meet the strict tolerance 10^{-12} we need to use $m = 25$, i.e. a method of order 51 around the shock.

5 Hermite-Taylor methods for systems in multiple dimensions

As a concrete example of a system of PDEs in multiple dimensions we consider Maxwell's equations in transverse magnetic form

$$\mu H_t^x = -E_y^z, \quad \mu H_t^y = E_x^z, \quad \varepsilon E_t^z = H_x^y - H_y^x, \tag{32}$$

on a rectangular domain $(x,y) \in [x_l, x_r] \times [y_b, y_t]$. To illustrate how simple boundary conditions can be imposed by a mirroring principle we consider the case where the boundary is a perfect electric conductor, i.e. $E_z = 0$. Then from (32) it is clear that that $H^x = 0$ and $H_x^y = 0$ on $x = x_l, x_r$ and $H^y = 0$ and $H_y^x = 0$ on $y = y_b, y_t$.
The discretization of (32) is a direct generalization of the one dimensional Hermite-Taylor method on a staggered grid consisting of a primal grid:

$$(x_i, y_j) = (x_l + ih_x, y_b + jh_y), \ (i,j) \in [0, n_x] \times [0, n_y], \ h_x = (x_r - x_l)/n_x, \ h_y = (y_t - y_b)/n_y,$$

and a dual grid

$$(x_{i+1/2}, y_{j+1/2}) = (x_l + (i+1/2)h_x, y_b + (j+1/2)h_y), \ i = 0, \ldots, n_x - 1, \ j = 0, \ldots, n_y - 1.$$

The method starts with the tensor product polynomials

$$u_{i,j,k_{\text{var}}}(x,y,t_0) = \sum_{l_x=0}^{m} \sum_{l_y=0}^{m} c_{l_x,l_y,k_{\text{var}}} \left(\frac{x - x_i}{h_x} \right)^{l_x} \left(\frac{y - y_j}{h_y} \right)^{l_y}, \tag{33}$$

where $u_{i,j,k_{\text{var}}}(x,y,t_0)$, $k_{\text{var}} = 1,2,3$ approximate H^x, H^y and E^z.
As in one dimension, the first step in the method is to form the Hermite interpolant at a dual node

$$u_{i+\frac{1}{2},j+\frac{1}{2},k_{\text{var}}}(x,y,t_0) = \sum_{l_x=0}^{2m+1} \sum_{l_y=0}^{2m+1} d_{l_x,l_y,k_{\text{var}}} \left(\frac{x - x_i}{h_x} \right)^{l_x} \left(\frac{y - y_j}{h_y} \right)^{l_y}. \tag{34}$$

Algorithmically, these polynomials are formed by applying the one dimensional interpolation to all y-derivatives at the bottom and top of a cell (see `get_tcofs1_2D`). and interpolating the resulting x-derivatives to the cell center. The interpolated data is evolved by the Taylor series technique and the time derivative coefficients are computed in `Maxwell2D_PDE`. At the end of the first half time step the solution is known on all the dual nodes inside the boundary. Evolution of the approximate solution at the primal nodes inside the boundary is carried out as described above. At the primal nodes on the boundary we form the Hermite interpolating polynomial by first extending the solution from interior dual nodes to ghost nodes just outside the boundary. The extension is done in such a way that the resulting interpolant is even or odd (depending on the boundary conditions, see `Maxwell2D_PDE`).

Table 5 Errors and convergence rates for the Maxwell TM cavity problem with $\omega_x = 4\pi$, $\omega_y = 8\pi$.

$m=5$	h_x	1.0	6.7(-1)	5.0(-1)	4.0(-1)	$m=10$	1.0	6.7(-1)	5.0(-1)	4.0(-1)
	l_2-error	2.1(0)	3.3(-1)	1.8(-2)	3.4(-3)		9.6(-4)	3.2(-7)	8.8(-10)	1.1(-11)
	Rate		4.5	10.1	7.5			19.8	20.5	19.7

To demonstrate the method we set $\mu = 1$ and $\varepsilon = 1$, then in the cavity $(x,y) \in [-1,1]^2$ a solution to the TM problem is

$$H^x = -\omega_y/\omega_t \sin(\omega_x x)\cos(\omega_y y)\sin(\omega_t t), \tag{35}$$
$$H^y = \omega_x/\omega_t \cos(\omega_x x)\sin(\omega_y y)\sin(\omega_t t), \tag{36}$$
$$E^z = \sin(\omega_x x)\sin(\omega_y y)\sin(\omega_t t), \tag{37}$$

with $\omega_t = \sqrt{\omega_x^2 + \omega_y^2}$.
We use this solution as initial data and evolve until time 3 and measure the error in the l_2-norm. As we choose $2\omega_x = \omega_y$ we also use $2h_x = h_y$ and set the time step as `dt = CFL*min(hx,hy)` with `CFL = 0.9`. The results, listed in Table 5, show a rate of convergence almost at the design rate.

6 Extensions and other work

Simulations of Compressible Flows: The first steps in constructing a compressible Navier-Stokes solver appear in the thesis of Dodson [15]. More recently we have been using the method to simulate compressible mixing layers, with an eye towards applications in aeroacoustics [17, 2, 1, 19].

Adaptive Implementations: The ease of incorporating p-adaptivity in Hermite methods is another of its attractive features, with the basic idea in one and two space dimensions explored in [8] and illustrated in the example above. We have also carried out preliminary studies of an h-adaptive version in [1]. Here we advocate quadtree/octree refinement of the Hermite cells with local time stepping. We believe that the stability of the resulting method follows directly from dissipativity of piecewise Hermite interpolation.

Dispersion and Dissipation: The dispersion and dissipation properties of Hermite methods in one and two space dimensions are studied in [3, 19, 20]. A conclusion is that the method is quite competitive in terms of cost with other high-order structured grid discretizations, particularly if large time steps are taken. Note that the primary errors are dissipation errors which occur when the the data is interpolated. Thus the method is most accurate (and most efficient) if the global time step is taken to be as large as possible while maintaining stability. Thus in the Runge-Kutta framework we suggest using as many substeps as needed to maintain accuracy and stability with a large global step.

Coupling with Other Methods: A drawback of Hermite methods is their reliance on structured grids and the need to utilize the PDE and geometry description to derive equations for normal derivatives at boundaries. To make the method more flexible we have implemented coupled Hermite-DG solvers on hybrid structured-unstructured grids [7]. Here the DG method obtains fluxes from the solution in neighboring Hermite cells, while Hermite cells bordering DG elements obtain data by interpolation. We adapt the local time stepping to

the requirements of each method, so at high order we take many steps within the DG elements for each Hermite step. Using dissipative upwind DG schemes we have experimentally found the method to be quite robust.

Of course we are not the only researchers to have used Hermite interpolation to solve differential equations. Hermite-based finite element methods have been studied for quite some time, in particular for problems posed in spaces H^2 or higher [10]. Among the first applications to hyperbolic equations can be found in the work of Yabe and collaborators [27]. More recently, Nave, Rosales, and Seibold have used Hermite interpolation to solve advection problems, with a particular interest in using the Hermite-based advection solver in conjunction with level set methods [23, 25, 9]. They term the methods jet schemes borrowing terminology from differential geometry. These methods differ from the one presented here in that a staggered mesh is not employed.
Hermite interpolation has also been proposed by Butcher as a way to construct SDIRK methods for solving stiff systems of ordinary differential equations [6]. The methods are explicitly interpreted as collocation methods employing the Hermite interpolant by Mülthei [22].
To conclude we recall a quote from Davis [13]: "Hermite's formulas are rediscovered and republished every four years." We hope we have demonstrated to the reader the unique and useful properties of Hermite interpolants and their potential use for solving differential equations. We also wish to encourage the use of the codes in `chides.org` and invite any feedback for their improvement.

7 Acknowledgements

Work of the first author was supported in part by ARO Grant W911NF-09-1-0344 and NSF Grants DMS-1418871, OCI-0904773. He also acknowledges the hospitality of the Courant Institute, where he was visiting during the preparation of the manuscript. Work of the second author was supported in part by NSF Grant DMS-1319054. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Army Research Office or the National Science Foundation.

References

1. D. Appelö, T. Colonius, M. Inkman, and T. Hagstrom. Recent progress on Hermite methods in aeroacoustics. In *17th AIAA/CEAS Aeroacoustics Conference*. AIAA, 2011.
2. D. Appelö and T. Hagstrom. Experiments with Hermite methods for simulating compressible flows: Runge-Kutta time-stepping and absorbing layers. In *13th AIAA/CEAS Aeroacoustics Conference*. AIAA, 2007.
3. D. Appelö and T. Hagstrom. On advection by Hermite methods. *Pacific J. Appl. Math.*, 4:125–139, 2012.

4. G. Birkhoff, M. Schultz, and R. Varga. Piecewise Hermite interpolation in one and two variables with applications to partial differential equations. *Numer. Math.*, 11:232–256, 1968.
5. P. Borwein and T. Erdélyi. *Polynomials and Polynomial Inequalities*. Springer-Verlag, New York, 1995.
6. J. C. Butcher. A generalization of singly-implict formulas. *BIT*, 21:175–189, 1981.
7. R. Chen, D. Appelö, and T. Hagstrom. A hybrid Hermite - discontinuous Galerkin method for hyperbolic systems with application to Maxwell's equations. *J. Comput. Phys.*, 257:501–520, 2014.
8. R. Chen and T. Hagstrom. P-adaptive Hermite methods for initial value problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46:545–557, 2012.
9. P. Chidwagyai, J.-C. Nave, R. Rosales, and B. Seibold. A comparative study of the efficiency of jet schemes. *Int. J. Numer. Anal. Model.-B*, 3:297–306, 2012.
10. P. Ciarlet. *The finite element method for elliptic problems*. Classics in Applied Mathematics 40. SIAM, Philadelphia, 2002.
11. G. Cohen. *Higher-Order Numerical Methods for Transient Wave Equations*. Springer-Verlag, New York, 2002.
12. G. Dahlquist and A. Björk. *Numerical Methods in Scientific Computing*, volume I. SIAM, Philadelphia, 2008.
13. P. Davis. *Interpolation and Approximation*. Dover Publications, New York, 1975.
14. Z. Ditzian. Multivariate Bernstein and Markov inequalities. *J. Approx. Theory*, 70:273–283, 1992.
15. C. Dodson. A High-Order Hermite Compressible Navier-Stokes Solver. Master's thesis, The University of New Mexico, 2003.
16. J. Goodrich, T. Hagstrom, and J. Lorenz. Hermite methods for hyperbolic initial-boundary value problems. *Math. Comp.*, 75:595–630, 2006.
17. T. Hagstrom, J. Goodrich, and G. Zhu. A Hermite-Taylor algorithm for simulating subsonic shear flows. In *12th AIAA/CEAS Aeroacoustics Conference*. AIAA, 2006.
18. J. Hesthaven and T. Warburton. *Nodal Discontinuous Galerkin Methods*. Number 54 in Texts in Applied Mathematics. Springer-Verlag, New York, 2008.
19. C.-Y. Jang, D. Appelö, T. Hagstrom, and T. Colonius. An analysis of dispersion and dissipation properties of Hermite methods and its application to direct numerical simulation of jet noise. In *18th AIAA/CEAS Aeroacoustics Conference*. AIAA, 2012.
20. C. Y. Jang and T. Hagstrom. An analysis of the dispersion and dissipation properties of Hermite methods. In preparation, 2014.
21. D. Kosloff and H. Tal-Ezer. A modified Chebyshev pseudospectral method with an $o(n^{-1})$ time step restriction. *J. Comput. Phys.*, 104:457–469, 1993.
22. H. N. Mülthei. Maximale konvergenzordnung bei der numerischen lösung von anfangswertproblemen mit splines. *Numer. Math.*, 39:449–463, 1982.
23. J.-C. Nave, R. Rosales, and B. Seibold. A gradient-augmented level set method with an optimally local, coherent advection scheme. *J. Comput. Phys.*, 229:3802–3827, 2010.
24. R. Neidinger. Efficient recurrence relations for univariate and multivariate Taylor series coefficients. *J. Amer. Inst. of Math. Sci.*, 2013:587–596, 2013.
25. B. Seibold, R. Rosales, and J.-C. Nave. Jet schemes for advection problems. *Discrete Contin. Dyn. Syst. Ser. B*, 17:1229–1259, 2012.
26. T. Warburton and T. Hagstrom. Taming the CFL number for discontinuous Galerkin methods on structured meshes. *SIAM J. Num. Anal.*, 46:3151–3180, 2008.
27. T. Yabe, T. Ishikawa, P. Wang, T. Aoki, Y. Kadota, and F. Ikeda. A universal solver for hyperbolic equations by cubic-polynomial interpolation. II. Two- and three-dimensional solvers. *Comput. Phys. Comm.*, 66:233–242, 1991.