Commun. Comput. Phys.
doi: 10.4208/cicp.080815.211116a

Vol. **21**, No. 5, pp. 1207-1230
May 2017

An Explicit Hermite-Taylor Method for the Schrödinger Equation

Daniel Appelö[1,*], Gunilla Kreiss[2] and Siyang Wang[2]

[1] *Department of Mathematics and Statistics, University of New Mexico, 1 University of New Mexico, Albuquerque, NM 87131, USA.*
[2] *Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, 75105 Uppsala, Sweden.*

Communicated by Chi-Wang Shu

Received 8 August 2015; Accepted (in revised version) 21 November 2016

Abstract. An explicit spectrally accurate order-adaptive Hermite-Taylor method for the Schrödinger equation is developed. Numerical experiments illustrating the properties of the method are presented. The method, which is able to use very coarse grids while still retaining high accuracy, compares favorably to an existing exponential integrator – high order summation-by-parts finite difference method.

AMS subject classifications: 65L20, 81V10

Key words: Hermite method, Schrödinger equation, high order.

1 Introduction

The quantum state of a physical system is described by the time dependent Schrödinger equation, which can only be solved exactly in very simplified settings. For many realistic problems the only alternative is to find approximate solutions by numerical methods. Challenges in designing numerical methods for the Schrödinger equation include: the exponential growth of the computational work with dimensionality, accurate propagation of dispersive waves and the parabolic–type time step constraint resulting form the second derivative.

For high-dimensional problems the computational cost associated with the high dimensionality can be reduced by the use of adaptive methods, especially if the solution is localized in space. To capture the dispersive properties of the solution high order accurate methods are preferred, in particular the Fourier pseudospectral method has been

*Corresponding author. *Email addresses:* appelo@math.unm.edu (D. Appelö), gunilla.kreiss@it.uu.se (G. Kreiss), siyang.wang@it.uu.se (S. Wang)

 ©2017 Global-Science Press

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

commonly used for the spatial discretization of the Schrödinger equation. Fourier based methods are spectrally accurate and capture the dispersion relation correctly, but the approximation is global making adaptive implementations all but impossible. Alternatively, high order finite difference methods, [13], which has the advantage of locality can be used. Even though high order of accuracy is easily attained in the interior of the computational domain, near the boundaries lower order accurate stencils are typically used to maintain stability. Other "method-of-lines" methods used to discretize the Schrödinger equation in space include the finite element method [9] and the Galerkin radial basis function method [10].

For most method-of-lines discretizations the parabolic time step restriction rules out the use of traditional explicit single-step and multi-step methods. Implicit single-step and multi-step methods can overcome the parabolic time step restriction but at the (often prohibitive) cost of having to solve linear systems of equations. As the equations are linear the semi-discretization takes the form of a system of first order linear ODE, $u_t = Au$, and the solution can be evolved exactly by exponentiating the matrix A. This approach is attractive as it is explicit but, being exact, does not suffer from the time step constraint. The cost of computing the matrix exponential is however large and often this type of exponential integrator method is combined with Krylov subspace methods to accelerate the computation of matrix exponential-vector products.

Aside from the absence of having to solve a large system of linear equations, required by most implicit methods, a main advantage of an explicit method is the ease of parallelization. However as previously eluded to, in order for an explicit method to be competitive it must be able to use very large cells or elements so that the parabolic time step constraint $\Delta t \leq Ch^2$, with h being a typical element size, is not overly restrictive. Local and spectrally accurate polynomial based methods such as spectral elements, discontinuous Galerkin or Hermite-Taylor methods, which are able to increase the size of the elements while increasing the polynomial degree to keep the accuracy constant, have the potential to operate in this regime. Here we develop an explicit method based on Hermite interpolation in space and evolution in time via Taylor series. This Hermite-Taylor method is spectrally accurate in space and time and, as we will show, is highly accurate on very coarse grids and with large time steps.

Hermite-Taylor methods were introduced by Hagstrom and coauthors in [5] and has since been used to solve different hyperbolic problems, see for example [1, 4] for some applications and [6] for a recent review. One of the most important features of Hermite-Taylor methods for hyperbolic problems is the ability to march the solution with a time step as large as allowed by the domain of dependence of the continuous problem. This is in stark contrast to most other polynomial based methods for hyperbolic problems, like spectral elements or discontinuous Galerkin methods. For these latter methods the time step has to be reduced by a factor of n^2 (n being the degree of the approximating polynomial) when a time- stepper with fixed order is used, and with a factor n when the order of the time-stepper is matched with the degree of the polynomials. This remarkable property of Hermite methods is rooted in the fact that Hermite methods only sample the

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

derivatives of the polynomials at the cell center where the derivative scales linearly with the degree n of the polynomial while it scales as n^2 at the edges of the cell.

The situation is even worse for higher derivatives, for example the rth derivative of a Chebyshev polynomial evaluated at the endpoints is

$$\frac{d^r}{dx^r}T_n(\pm)=(\pm)^{n+r}\prod_{k=0}^{r-1}\frac{n^2-k^2}{2k+1}.$$

For $r=2$, relevant in the present context, we get $|\frac{d^2}{dx^2}T_n(\pm)|\sim n^4$, to be compared to $|\frac{d^2}{dx^2}T_n(0)|\sim n^2$, at the cell center. Thus, our method has the potential to take n^2 times larger time steps than explicitly time-stepped spectral elements or discontinuous Galerkin methods. Below we will use polynomials of degree as high as 33 where the gain in efficiency could be as large as three orders of magnitude.

The rest of the paper is organized as follows. In Section 2 we describe the spatial discretization by Hermite interpolation and the time evolution of the approximation via Taylor time series. We also discuss how to handle boundary conditions and variable coefficients, and how to incorporate order adaptivity. In Section 3 we present numerical experiments. We begin by illustrating the time-stepping properties of the method and demonstrate its accuracy, especially on coarse grids. We then compare the methods efficiency with a high order summation-by-parts discretization evolved using an exponential integrator relying on a Lanczos procedure. We also demonstrate the performance of our order-adaptive method and present experiments in two dimensions. Section 4 summarizes the paper and presents possible extensions of the method.

2 Description of the method

In this paper we are concerned with finding approximate solutions to the Schrödinger equation, which in one dimension and for a single particle of mass M in a potential $V(x)$ can be written as

$$\begin{aligned} &i\hbar\frac{\partial u(x,t)}{\partial t}=-\frac{\hbar^2}{2M}\frac{\partial^2 u(x,t)}{\partial x^2}+V(x)u(x,t),\quad t>0,\quad x_l\le x\le x_r,\\ &u(x,0)=u_0(x). \end{aligned} \tag{2.1}$$

Here the constant $\hbar$ is the reduced Planck's constant. After standard non-dimensionalization we can reformulate (2.1) into

$$i\frac{\partial u(x,t)}{\partial t}=-\frac{\partial^2 u(x,t)}{\partial x^2}+V(x)u(x,t),\quad t>0,\quad x_l\le x\le x_r. \tag{2.2}$$

To discretize Eq. (2.2) we introduce two Cartesian grids, a primal grid

$$x_j=x_l+jh_x,\quad j=0,\cdots,N_x, \tag{2.3}$$

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

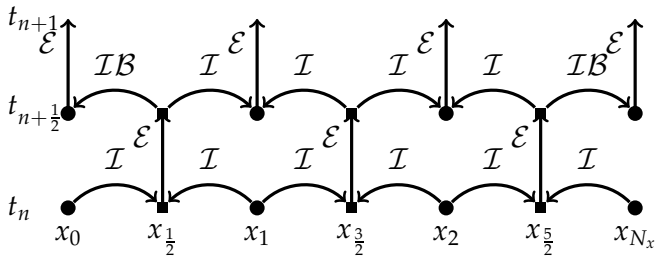


Figure 1: Schematic description of the algorithm. The primal grid is denoted by circles and the dual grid is denoted by squares. For the first half time step we first construct the unique degree $2m+1$ polynomial which (Hermite) interpolates two degree m polynomials associated with two adjacent primal grid points. The process is denoted by $\mathcal{I}$ in the figure. The new polynomial is then evolved for a half time step and then projected (by simply removing the higher degree terms) into a degree m polynomial. This process is denoted by $\mathcal{E}$ in the figure. The second half step proceeds in the same way as the first half step but the interpolation uses the polynomials centered at the dual grid points that has just been updated and centers the new degree $2m+1$ polynomials at the primal grid points. The degree $2m+1$ polynomials at the first and last primal grid points are constructed from the nearby interior polynomials together with the boundary conditions and the PDE. In the picture this interpolation process is denoted $\mathcal{IB}$.

and a dual grid

$$x_j = x_l + jh_x, \quad j = \frac{1}{2}, \cdots, N_x - \frac{1}{2}, \tag{2.4}$$

where

$$h_x = \frac{x_r - x_l}{N_x}.$$

See Fig. 1 for a schematic of the grids.

We begin by approximating the solution $u(x,t)$ around each primal grid point at time t_n, $(x,t) = (x_j, t_n)$, by a degree m polynomial

$$p_j(x,t_n) = \sum_{l=0}^{m} c_{l0}[x_j](x - x_j)^l. \tag{2.5}$$

The degrees of freedom in the method are thus the $(m+1)\times(N_x+1)$ coefficients

$$c_{l0}[x_j], \quad l = 0, \cdots, m, \quad j = 0, \cdots, N_x. \tag{2.6}$$

Note that, as the approximation is a truncated Taylor series expansion we also have the following relation between the coefficients $c_{l0}[x_j]$ and the solution u and its spatial derivatives at x_j

$$c_{l0}[x_j] \approx \frac{1}{l!}\frac{\partial^l u(x_j,t_n)}{\partial x^l}, \quad l = 0, \cdots, m. \tag{2.7}$$

Thus, the degrees of freedom in a Hermite method can also be thought of as the solution and its m first (scaled) derivatives at a grid point. We can thus think of the method as an element based method with elements consisting of the interval in-between two primal

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

(or dual at the next half time step) grid points or as a nodal based method with multiple degrees of freedom associated with each primal grid point.

At the start of the computation we find the coefficients (2.6) from the initial data, either by symbolically computing the derivatives of the initial data or by some sufficiently accurate approximation (for example by standard interpolation).

At a first glance it may appear cumbersome to compute higher order derivatives of the initial data, however for most known functions[†] it is possible to find stable and concise recursion relations for the derivatives. For example the Gaussian $f(x)=\exp(-\sigma x^2)$ has the recursion

$$f(x)=f(x),\quad \frac{df(x)}{dx}=-2\sigma x f(x), \tag{2.8}$$

$$\frac{d^{p+1}f(x)}{dx^{p+1}}=-2\sigma\left(x\frac{d^p f(x)}{dx^p}+(p-1)\frac{d^{p-1}f(x)}{dx^{p-1}}\right),\quad p=1,2,\cdots. \tag{2.9}$$

2.1 Hermite interpolation

With the coefficients (2.7) known we use the approximate solutions from two adjacent grid points x_j and x_{j+1} to construct the unique Hermite interpolating polynomial centered at the dual node $x_{j+1/2}$. Precisely, we find the unique degree $2m+1$ polynomial

$$p_{j+1/2}(x,t_n)=\sum_{l=0}^{2m+1}c_{l0}[x_{j+1/2}](x-x_{j+1/2})^l, \tag{2.10}$$

which satisfies the $2\times(m+1)$ interpolation conditions

$$\frac{d^l p_{j+1/2}(x_j,t_n)}{dx^l}=\frac{d^l p_j(x_j,t_n)}{dx^l},\quad \frac{d^l p_{j+1/2}(x_{j+1},t_n)}{dx^l}=\frac{d^l p_{j+1}(x_{j+1},t_n)}{dx^l}, \tag{2.11}$$

for $l=0,\cdots,m$. In the lowest order case $m=0$ this process is simply a linear interpolation of two point values. For $m=1$ the Hermite interpolant is a cubic polynomial created from two first degree polynomials etc.

Forming (2.10) is equivalent to the unique one-to-one mapping of the coefficients from the polynomials at x_j and x_{j+1} into the $2m+2$ coefficients in (2.10). Algorithmically this is done either by assembling the coefficients on the primal grid points into a column vector

$$c=\begin{bmatrix} c_{00}[x_j] \\ c_{10}[x_j], \\ \vdots \\ c_{m0}[x_j] \\ c_{00}[x_{j+1}] \\ \vdots \\ c_{m0}[x_{j+1}] \end{bmatrix},$$

[†]The fact is that most known functions were actually introduced as solutions to special ODE. Trigonometric functions, Bessel functions and exponentials are some examples.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

which is multiplied with a pre-computed $(2m+2)\times(2m+2)$ matrix, yielding the coefficients of (2.10). In what follows we refer to this as the direct mapping approach. Alternatively, we first form a generalized Newton table where the coefficients of the Newton form of the Hermite interpolating polynomial can be obtained from the diagonal of the table. Then we use a fast dual-Vandermonde solve to find the coefficients for the Taylor form used above (see [6] for details).

2.2 Time evolution

The polynomial (2.10) is the spatial approximation which will be evolved in time using the governing PDE. We start by expanding $p_{j+1/2}(x,t_n)$ in time by a Taylor series (the upper limit $q(m,l)$ in the second sum is dictated by accuracy requirements discussed below)

$$p^n_{j+1/2}(x,t)=\sum_{l=0}^{2m+1}\sum_{s=0}^{q(m,l)}c_{ls}[x_{j+1/2}](x-x_{j+1/2})^l(t-t_n)^s. \tag{2.12}$$

The approximation to the solution and the (scaled) derivatives of the solution on the dual grid at the next half time step will simply be (2.12) evaluated at $(x_{j+1/2},t_{n+1/2})$ and truncated at degree m. That is:

$$\frac{1}{l!}\frac{\partial^l u(x_{j+1/2},t_{n+1/2})}{\partial x^l}\approx\sum_{s=0}^{q-l}c_{ls}[x_{j+1/2}]\left(\frac{\Delta t}{2}\right)^s,\quad l=0,\cdots,m. \tag{2.13}$$

Again, note that we only keep the $m+1$ first coefficients as the next half time step will proceed analogously to the first half step and we thus only need m derivatives (or equivalently $m+1$ coefficients).

However, before (2.12) can be evaluated the coefficients in the expansion (we currently only know the $s=0$ terms from the initial data) must be found. To find these we will use the governing PDE.

To see how the coefficients c_{ls}, with $s>0$ can be related to c_{l0} by using the PDE we first consider the special case $V(x)=0$. For this case the PDE is reduced to

$$\frac{\partial u(x,t)}{\partial t}=i\frac{\partial^2 u(x,t)}{\partial x^2}. \tag{2.14}$$

Now, assuming that the solution is smooth enough, we take time and space derivatives of the equation and replace $u(x,t)$ by $p^n_{j+1/2}(x,t)$, obtaining

$$\frac{\partial^{r+1+k}}{\partial t^{r+1}\partial x^k}p^n_{j+1/2}(x,t)=i\frac{\partial^{r+k+2}}{\partial t^r\partial x^{k+2}}p^n_{j+1/2}(x,t). \tag{2.15}$$

Evaluating this equation at $(x_{j+1/2},t_n)$ we obtain (suppressing $[x_{j+1/2}]$)

$$(s+1)!l!c_{l,s+1}=is!(l+2)!c_{l+2,s}. \tag{2.16}$$

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

After some rearrangements and shifting of the s index, the recursion takes the simple form

$$c_{l,s}=i\frac{(l+2)(l+1)}{s}c_{l+2,s-1},\quad l=0,\cdots,2m+1,\quad s=1,\cdots,q(m,l). \tag{2.17}$$

When $V(x)\neq 0$ the recursion becomes slightly more involved. Here we assume that the potential has been expanded as a *local* degree $2m+1$ polynomial around each grid point. Then the equation contains a product of two polynomials and the recursion becomes

$$c_{ls}=i\frac{(l+2)(l+1)}{s}c_{l+2s-1}-\sum_{i=0}^{l}V^{[l-i,0]}c_{is-1},\quad l=0,\cdots,2m+1,\quad s=1,\cdots,q(m,l). \tag{2.18}$$

Here the notation

$$Q^{[k,l]}(x,t)=\frac{1}{k!l!}\frac{\partial^{k+l}Q}{\partial x^k\partial t^l}(x,t),$$

is used for scaled derivatives.

To understand how the product, $V(x)u(x,t)$, is computed in our implementation it is useful to recall that $V(x)$ and $u(x,t)$ are approximated by two polynomials, say

$$q_V(x)=\sum_{l=0}^{2m+1}d_l(x-x_{j+1/2})^l$$

and

$$p(x;t)=\sum_{l=0}^{2m+1}c_l(t)(x-x_{j+1/2})^l.$$

Here we keep the approximation semi-discrete for brevity. We momentarily ignore the u_{xx} term, so the equation becomes

$$u_t=-iV(x)u(x,t). \tag{2.19}$$

Replacing u and V by their approximations and taking spatial derivatives of the equation yields

$$\frac{\partial^k}{\partial x^k}\sum_{l=0}^{2m+1}c_l'(t)(x-x_{j+1/2})^l=-i\frac{\partial^k}{\partial x^k}\left[\sum_{l=0}^{2m+1}d_l(x-x_{j+1/2})^l\sum_{l=0}^{2m+1}c_l(t)(x-x_{j+1/2})^l\right]. \tag{2.20}$$

Now, if we denote by q_{Vu} the degree $2m+1$ truncated polynomial

$$q_{Vu}(x;t)=\mathcal{T}^{2m+1}(q_V(x)p(x))=\sum_{l=0}^{2m+1}b_l(t)(x-x_{j+1/2})^l,$$

and evaluate (2.20) at $x=x_{j+1/2}$ we find

$$c_k'(t)=b_k(t),\quad k=0,\cdots,2m+1,$$

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

where the prime denotes derivative with respect to time. Thus, with the $V(x)u(x)$ term we may first find all l coefficients for $s=1$ by performing a truncated multiplication of two polynomials. Then for higher s we may differentiate the above equation and repeat the procedure.

2.2.1 Truncation of the time-Taylor series

Having discussed the recursion relations for the coefficients in the space-time polynomial (2.12) we now consider the truncation $q(m,l)$ of its temporal expansion.

The truncation error of the approximation (2.12) in $(x,t)\in[x_j,x_{j+1}]\times[t_n,t_{n+1/2}]$ will be bounded by

$$\|u(x,t)-p^n_{j+1/2}(x,t)\|_\infty \le C\left(h_x^{(2m+1)+1}\Delta t^{q(m,2m+1)+1}+\cdots+h\Delta t^{q(m,0)+1}\right). \tag{2.21}$$

Anticipating a parabolic stability restriction $\Delta t\le C(m)h_x^2$ we thus choose

$$q(m,l)=m-\lfloor l/2\rfloor, \tag{2.22}$$

resulting in a local truncation error of order h_x^{2m+2}. Here $\lfloor\cdot\rfloor$ is the floor operator. As we are time-stepping the solution in an explicit fashion we will need to take $N_t\sim\frac{1}{h_x^2}$ time steps resulting in a global error of order h_x^{2m}. We note that in the absence of a potential the temporal Taylor series truncates with the choice $q(m,l)=m-\lfloor l/2\rfloor$.

2.2.2 Completion of a full time step

As mentioned above, once the coefficients have been computed the solution at the dual grid point $x=x_{j+1/2}$ can be obtained at $t=t_{n+1/2}$ from (2.13). With $m+1$ coefficients known on the dual grid, the same procedure is carried out for the solution at all dual grid points to produce the solution at the primal grid points at the next half time step $t=t_{n+1}$. See also Fig. 1. This concludes a full time step.

Remark 2.1. Note that the evolution process is purely local on each element and thus allows for straightforward implementation on a parallel computer. Also, the storage requirements are optimal as only a single local copy of the $(2m+2)\times q$ doubles c_{ls} is required. Comparing with a q-stage Runge-Kutta – "method of lines" method, the memory savings is at least a factor of q.

2.3 Boundary conditions

As half of the degrees of freedom on the cell next to the boundary located on the boundary we must provide $m+1$ boundary conditions to fully specify the Hermite interpolant in that cell. The "extra" m boundary conditions are typically derived by using the PDE and tangential derivatives of the PDE together with the boundary condition. However if the boundary is curved or if the boundary conditions are complex this may be rather

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

involved and the use of a hybrid discontinuous Galerkin-Hermite approach, as described for Maxwell's equations in [4], can be used.

In this work we limit ourselves to periodic and homogenous Dirichlet boundary conditions. The latter is often used in situations where the solution is guided by a potential, resulting in the solution being very small at the edge of the computational domain. In such situations we can simply set the solution and all the derivatives to zero at the boundary. For situations when the solution is not small but the boundary is planar we may simply use an image principle to construct the interpolant satisfying the boundary conditions and the PDE at the boundary, see also [6].

2.4 Adaptive implementation

As was shown for hyperbolic problems by Chen and Hagstrom, [3], Hermite methods are well suited for order or p-adaptive implementations. In this section we describe how the p-adaptive method by Chen and Hagstrom can be applied to the Schrödinger equation.

The modification to the basic Hermite-Taylor method is minimal due to the locality of the method. As before we consider the approximation of the solution u around some grid point x_j to be a polynomial of degree m_j, but we now allow for the degree to change between grid points. The time-evolution starts by forming the interpolant centered around a dual grid point $x_{j+1/2}$. Now, the highest possible degree this interpolant can have is $m_j+m_{j+1}+1$ but as in [3] we find that the method is more robust if we use the degree $\bar{m}=\min(m_j,m_{j+1})$ polynomials at x_j and x_{j+1} to construct the interpolant at $x_{j+1/2}$.

Once the coefficients in the Taylor series has been computed using one of the recursions (2.17), (2.18) the order adaptive step amounts to deciding a suitable value for the degree of the new data at $(x_{j+1/2},t_{n+1/2})$. Assuming a tolerance TOL is required we then truncate at the smallest $m_{j+1/2}$ that satisfies

$$\max_{l>m_{j+1/2}} \frac{1}{l!}\left|\frac{\partial^l u(x_{j+1/2},t_{n+1/2})}{\partial x^l}\right| < \text{TOL}, \tag{2.23}$$

and is less than or equal to some upper a-priori bound on the maximal degree.

2.5 Multiple dimensions

The extension to multiple dimensions is straightforward. In two dimensions, we again introduce a primal

$$x_j = x_l + jh_x, \quad j=0,\cdots,N_x, \quad y_j = y_l + jh_y, \quad j=0,\cdots,N_y, \tag{2.24}$$

and a dual grid

$$x_j = x_l + jh_x, \quad j=\frac{1}{2},\cdots,N_x-\frac{1}{2}, \quad y_j = y_l + jh_y, \quad j=\frac{1}{2},\cdots,N_y-\frac{1}{2}, \tag{2.25}$$

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

where

$$h_x = \frac{x_r - x_l}{N_x}, \quad h_y = \frac{y_r - y_l}{N_y}.$$

The approximation on each primal node now consists of a tensor product polynomial

$$p_{j,k}(x,y,t_n) = \sum_{l_x=0}^{m} \sum_{l_y=0}^{m} c_{l_x,l_y,0}[x_j,y_k](x-x_j)^{l_x}(y-y_k)^{l_y},$$

which together with the data at the other three corners of a cell can be combined into a Hermite interpolant centered at the dual grid point

$$p_{j+\frac{1}{2},k+\frac{1}{2}}(x,y,t_n) = \sum_{l_x=0}^{2m+1} \sum_{l_y=0}^{2m+1} c_{l_x,l_y,0}[x_{j+\frac{1}{2}},y_{k+\frac{1}{2}}](x-x_{j+\frac{1}{2}})^{l_x}(y-y_{k+\frac{1}{2}})^{l_y}.$$

Algorithmically, forming the above Hermite interpolant is done by repeated use of the one-dimensional mapping, say in the y-direction, for the function and all the x-derivatives at two grid points (x_j,y_k) and (x_j,y_{k+1}) and (separately) at the two grid points (x_{j+1},y_k) and (x_{j+1},y_{k+1}). The result is two polynomials of degree $2m+1$ in y and m in x centered at $(x_j,y_{k+\frac{1}{2}})$ and $(x_{j+1},y_{k+\frac{1}{2}})$. These are again mapped with the one-dimensional procedure into the above Hermite interpolant.

As in the one dimensional case the time evolution is performed by Taylor series approximation where the higher order time derivatives are found from differentiation of the PDE. For example the PDE

$$u_t = i(u_{xx} + u_{yy}),$$

would yield the following recursion for the coefficients

$$c_{l_x,l_y,s} = i\left(\frac{(l_x+2)(l_x+1)}{s} c_{l_x+2,l_y,s-1} + \frac{(l_y+2)(l_y+1)}{s} c_{l_x,l_y+2,s-1}\right). \tag{2.26}$$

Generalizations to higher dimensions would be analogous.

The adaptive method is also easily extended. Denote the number of derivatives in the x and y direction at a point (x_j,y_k) by $m_{x,j,k}$ and $m_{y,j,k}$. The mappings in the y-direction is then performed using $\bar{m}_l = \min(m_{y,j,k}, m_{y,j,k+1})$ and $\bar{m}_r = \min(m_{y,j+1,k}, m_{y,j+1,k+1})$ and the final mapping in the x-direction is performed using $\bar{m} = \min(\bar{m}_l, \bar{m}_r)$, for details see [3].

3 Experiments

This section presents experiments illustrating the properties of the method. We begin by considering how the maximum allowable time step depends on the degree of the polynomials used in the approximations.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

3.1 Time step restrictions

To determine how the order of accuracy of the method impacts the time step we consider

$$u_t = iu_{xx}, \quad t>0, \quad -10\leq x\leq 10, \tag{3.1}$$

with periodic boundary conditions.

Let $U(t)$ be a vector containing all the degrees of freedom in the Hermite-Taylor method for the above equation. Then the solution after a full time step is $U(t+\Delta t) = AU(t)$ where A is a square matrix[‡] with the number of rows and columns equaling the number of degrees of freedom. In order for the method to be stable we require that the time step is small enough so that all the eigenvalues of A lie inside the unit disc. With the right hand side being a second derivative and with an explicit time integrator it is reasonable to assume a time step restriction on the form

$$\Delta t \leq C(m)h_x^2.$$

Before investigating how $C(m)$ depends on m we first consider the influence of the numerical conditioning of the interpolation process itself and how it depends on m. To do this we pick a very small $C=10^{-5}$ and compute the eigenvalues of A. With such a small C the method will be stable in exact arithmetic but due to finite precision effects the matrix A may still have a spectral radius, $\rho(A)$, larger than one. We thus discretize (3.1) using $m=1,\cdots,25$ and set $q=m$. For each discretization we compute the quantity $\kappa = |1-\rho(A)|$. Fig. 2 displays the results for the two different interpolation approaches and for two different number of grid points, $N_x = 20,40$.

As can be seen in the figure, the direct mapping approach is slightly better than the Newton approach but the conditioning for both approaches is independent of the number of grid points used. The independence of the discretization size is not unexpected as the interpolation is purely local on one cell at a time and is performed on a scaled interval $z=(x-x_{j+1/2})/h_x \in [-1/2,1/2]$.

Next we find $C(m)$ by performing a bisection search using the criterion $\kappa<1+10^{-8}$ to distinguish between a stable and unstable time step. Based on the conditioning of the interpolation we only consider $m=1,\cdots,16$. In Fig. 2 we display $m\Delta t/h_x^2$ for three different grid spacings, $N_x=20,40,60$. As can be seen the choice $C(m)=0.1/m$ will give a stable method for this problem. This scaling is in line with the discussion in the introduction.

Of course, if the governing equation contains a potential it may be that the scaling constant needs to be adjusted somewhat, but as we will see below the adjustments appear to be small for the problems we have considered.

[‡]The matrix is easily constructed one column at a time by taking one time step with unit vectors corresponding to the degrees of freedom as initial data.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

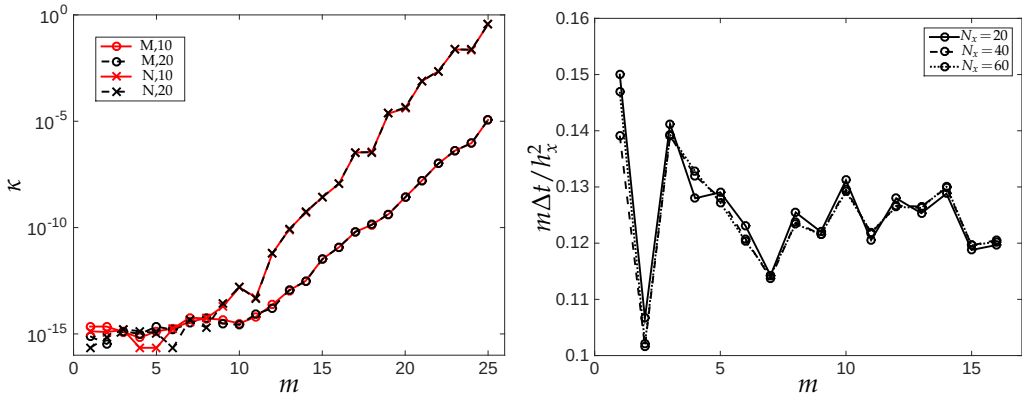


Figure 2: To the left: Distance between the unit circle and the eigenvalue with largest magnitude as a function of m. The label indicates the method and the number of grid-points used, e.g. $M10$ refers to direct mapping (see Section 2.1) with $N_x=10$. To the right: The largest value of $m\Delta t/h_x^2$, resulting in a stable method, as a function of m.

3.2 Evolution of a free particle

In this section we consider the evolution of a free particle and solve

$$u_t = iu_{xx}, \quad t>0, \quad x_l \le x \le x_r, \tag{3.2}$$

with initial data

$$u(x,0) = (\cos(k_0 x) + i\sin(k_0 x))e^{-x^2}. \tag{3.3}$$

The exact solution is

$$u(x,t) = \sqrt{\frac{i}{i-4t}}\exp\left(\frac{-ix^2 - k_0 x + k_0^2 t}{i-4t}\right),$$

which we use to impose boundary conditions and to compute errors.

3.2.1 Observed order of accuracy

We begin by experimentally determining the observed order of accuracy by evolving the initial data (3.3) on a domain with $x_l=-10$ and $x_r=10$. We choose $k_0=7$ and solve the equations up to time 0.4. The initial data and final solution are displayed in Fig. 3.

To check the order of accuracy we vary h_x and measure the l_2-error at the final time. The errors as a function of h_x are plotted in Fig. 4. Linear least squares fits for the model

$$\mathcal{E}(h_x) = Const \times h_x^p,$$

to the error corresponding to the three smallest h_x have been performed. The estimated order of accuracy p, reported in Table 1, agree reasonably well with the expected $p=2m$.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

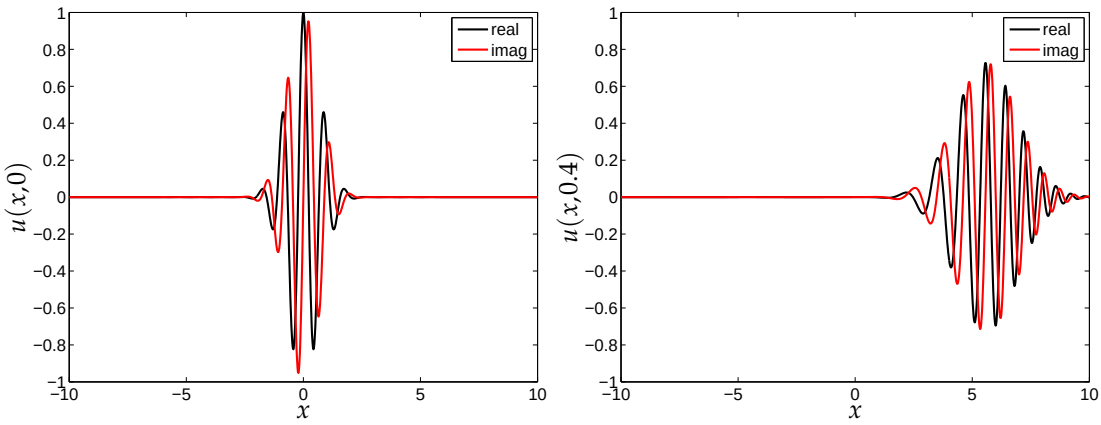


Figure 3: Initial (left) and final (right) solution.

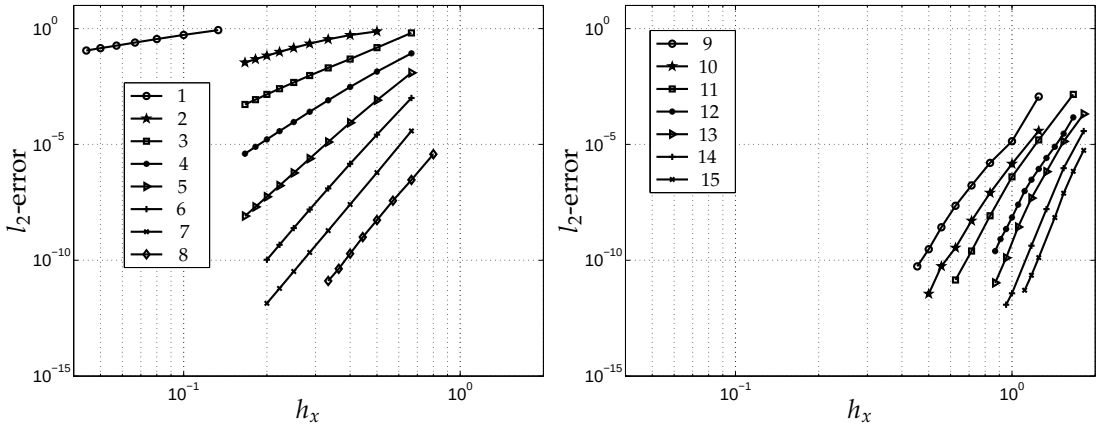


Figure 4: Errors in the l_2-norm as a function of h_x for methods with $m=1,\cdots,15$. The limits of the axis are the same in both figures so that they can be compared.

Table 1: Order of accuracy estimated from three smallest h_x. The results agree reasonably well with the expected relation $p=2m$.

m	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
p	2.0	3.7	5.5	7.8	10.5	14.0	14.2	15.2	19.1	19.8	21.9	23.7	28.1	28.5	28.9

3.2.2 Resolution on very coarse grids

As the suggested method is explicit we must demonstrate its resolving power for challenging problems on very coarse grids where we can take very large time steps. To do so we solve the same problem as above and discretize the domain with a large step size, $h_x=1$, corresponding to about one grid point per wavelength. With this extremely coarse resolution we evolve the equation until the final time, 0.4, when we record the maximum

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

error in the solution at the primal nodes (the initial and final solution are displayed in Fig. 3).

Due to the large h_x the number of time steps to reach the end time range from a handful for $m=1$ to about 50 for $m=16$. The results for different m are displayed in Table 2. As can be seen the highest order methods are capable of evolving the solution very accurately. It can also be noted that the error appears to be at its minima for $m=15$, for larger m finite precision effects will come into play.

Table 2: Maximum error at time 0.4 as a function of number of derivatives for the experiment with a free particle.

m	1	2	3	4	5	6	7	8
error	1.2(0)	6.9(-1)	7.0(-1)	3.1(-1)	1.2(-1)	2.7(-2)	3.9(-3)	2.2(-4)
m	9	10	11	12	13	14	15	16
error	1.0(-5)	9.8(-7)	2.7(-7)	5.0(-9)	4.6(-10)	2.4(-12)	4.1(-13)	1.1(-12)

To illustrate how coarse the grid is, Fig. 5 displays the piecewise linear interpolant of the solution obtained with $m=15$ along with the full piecewise interpolant evaluated on a much finer grid.

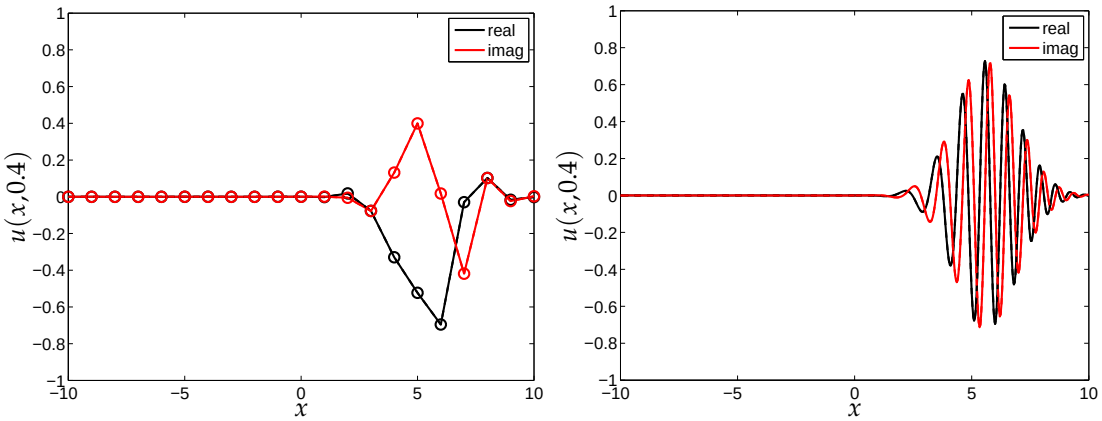


Figure 5: Numerical solution at the final time (using $m=15$). To the left is the piecewise linear solution obtained from the function values at the primal grid points (indicated by the circles). To the right the same numerical solution is interpolated (using all data) on a fine grid.

3.2.3 Efficiency

In order to demonstrate the efficiency of the method we have compared the Hermite-Taylor method with a summation-by-parts (SBP) [11] finite difference discretization coupled with the exponential integrator time-stepping method [8].

In this example we choose $x_l=-10$, $x_r=20$ and simulate up to time $t=0.4$. For this domain the solution is negligible at both boundaries and we may impose that the solution

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

and its derivatives are zero at the boundaries. On the primal grid (2.3), we introduce a grid function $v_j(t) \approx u(x_j,t)$ and set $v=[v_0,v_1,\cdots,v_{N_x}]^T$ to be the vector valued grid function. We also define an inner product and a norm for the complex vectors $a,b\in\mathbb{C}^{N_x+1}$ as $(a,b)_H=a^*Hb$ and $\|a\|_H^2=a^*Ha$, respectively, where $*$ denotes the conjugate transpose and H is a positive definite matrix.

As mentioned above the spatial discretization is performed by using finite difference operators that satisfy the summation by parts property. These operators mimic the integration by parts formula from the continuous setting via the associated norms. Precisely, let D be the discrete operator approximating the second derivative, i.e. $D\approx\frac{\partial^2}{\partial x^2}$, then it is a diagonal norm SBP operator if it can be written as $D=H^{-1}(-A+BS)$, where H is diagonal and positive definite, A is symmetric positive semidefinite and $B=diag(-1,0,\cdots,0,1)$. Here, S is a one sided approximation of the first derivative at the boundaries and H is the norm associated with D.

In [12], $2p^{th}$ $(p=1,2,3,4)$ order accurate diagonal norm SBP operators are constructed by using standard central finite differences in the interior and special one sided stencils near the boundaries. Though termed $2p^{th}$ order accurate, the approximation error of the second derivative is of order $2p$ in the interior and of order p near the boundaries, and the approximation error of the first derivative at the boundaries by the one sided operator S is of order $p+1$. When applied to the Schrödinger equation, these schemes typically are of order $\min(2p,p+2)$ [13].

An SBP operator itself does not impose any boundary condition. To guarantee strict stability, a common approach is to impose the boundary conditions weakly by the simultaneous approximation term (SAT) method [2]. The SAT acts as a penalty term that drags the numerical solution at the boundaries towards the boundary conditions. The semi-discrete approximation of Eq. (3.2) is

$$v_t=iDv+\tau_l H^{-1}S^Te_0e_0^Tv+\tau_r H^{-1}S^Te_{N_x}e_{N_x}^Tv, \tag{3.4}$$

where $e_0=[1,0,\cdots,0]^T$ and $e_{N_x}=[0,\cdots,0,1]^T$. τ_l and τ_r are penalty parameters to be determined so that (3.4) is stable. Multiplying (3.4) with v^*H and adding the conjugate transpose of the same equation we obtain

$$\begin{aligned}\frac{d}{dt}\|v\|_H^2&=(v,v_t)_H+(v_t,v)_H\\&=v^*Hv_t+v_t^*Hv\\&=(\tau_r^*+i)v^*e_{N_x}e_{N_x}^TSv+(\tau_r-i)v^*S^Te_{N_x}e_{N_x}^Tv\\&\quad+(\tau_l^*-i)v^*e_0e_0^TSv+(\tau_l+i)v^*S^Te_0e_0^Tv.\end{aligned}$$

By choosing $\tau_l=-i$ and $\tau_r=i$ we get the energy estimate $\frac{d}{dt}\|v\|_H^2=0$, which means that the discrete energy $\|v\|_H^2$ is conserved and (3.4) is stable. Let $Q=D-H^{-1}S^Te_0e_0^T+H^{-1}S^Te_{N_x}e_{N_x}^T$, then (3.4) can be written as

$$v_t=iQv. \tag{3.5}$$

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

The time-stepping of (3.5) with a standard explicit ODE solver requires that the time step scales as $\Delta t \sim h_x^2$. This will typically lead to a prohibitively small time step as the order of the SBP discretization is limited to eight, effectively forcing h_x to be small when small errors are desired. Therefore, a common approach to circumvent the small time step restriction is to use an exponential integrator method [8]. It is straightforward to verify from (3.5) that the matrix exponential $e^{iQ\Delta t}$ takes the numerical solution from t_n to t_{n+1} in the closed form

$$v(t_{n+1}) = e^{iQ\Delta t} v(t_n), \tag{3.6}$$

When the dimension of Q is large (as is often the case) it becomes infeasible to compute the matrix exponential directly. Instead, a Krylov subspace method is used to approximate the right hand side of (3.6) as

$$e^{iQ\Delta t} v(t_n) \approx V_r e^{iT_r\Delta t} e_1 \|v(t_n)\|_2, \tag{3.7}$$

where $V_r \in \mathbb{C}^{(1+N_x)\times r}$ is the orthonormal basis of the r^{th} order Krylov subspace $K_r(Q, v(t_n))$ for Q and $v(t_n)$, $T_r \in \mathbb{C}^{r\times r}$ is an upper Hessenberg matrix and e_1 is the first unit vector in $\mathcal{R}^r$. In practice $r \ll N_x$, which makes it inexpensive to compute $e^{iT_r\Delta t}$. For a general matrix Q, the orthonormal basis of $K_r(Q, v(t_n))$ is constructed by the Arnoldi method. If the matrix Q is Hermitian, the Arnoldi method is simplified to the much more efficient Lanczos method, with T_r being a tridiagonal matrix.

A direct discretization of (3.2) by the SBP-SAT method leads to a non-Hermitian matrix but by the coordinate transformation $w = H^{\frac{1}{2}} v$, the corresponding spatial discretization matrix is imaginary and symmetric, and the Lanczos method is applicable.

A larger order of the Krylov subspace allows for a longer time step but may induce a loss of orthogonality in the orthonormal basis. To maximize the computational efficiency, we choose the time step adaptively as follows.

In each time step, the size of the Krylov subspace r starts from 1 and increases by 1 in each iteration. When $r \geq 3$, the residual [7]

$$R_r = |\Delta t [e^{iT_r\Delta t}]_{r,1} (T_{r+1})_{r+1,r}| < tol, \tag{3.8}$$

is computed and compared with the tolerance tol. Here $[\cdot]_{r,1}$ denotes the $(r,1)$ entry of the matrix and $(T_{r+1})_{r+1,r}$ is the $(r+1,r)$ entry of T_{r+1}. If $R_r < tol$, then we stop the iteration and compute the result by (3.7); otherwise we continue the next iteration with $r := r+1$. If the residual is still larger than the tolerance when r reaches the predetermined upper limit (here chosen as 25), a smaller time step is used and the iteration is restarted. In this experiment, where the exact solution is known, we also choose the tolerance tol adaptively to match the error in the numerical solution. This maximizes the computational efficiency of the simulation.

To compare the Hermite method and the SBP-SAT-Lanczos method we have implemented both methods in MATLAB R2012b. The evolution of the free particle is simulated on a MacBook Pro with 2.9 GHz Intel Core i7 processor and 8 GB 1600 MHz DDR3

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

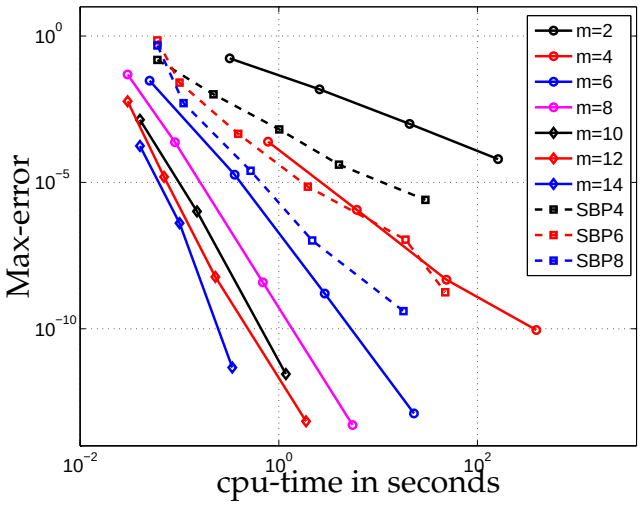


Figure 6: Efficiency comparison between Hermite-Taylor method and SBP-SAT-Lanczos finite difference method.

SDRAM. The timing of the run time is performed over the time-loop in the two different codes and the results are displayed in Fig. 6. Although results may differ slightly for another implementation and different hardware, it appears clearly that the high order Hermite-Taylor method is more efficient than the SBP-SAT-Lanczos method. For example, to achieve the maximum error 10^{-8}, Hermite-Taylor method with $m=15$ uses about $\frac{1}{50}$ CPU time as the 8^{th} order SBP-SAT-Lanczos scheme does.

An alternative measure of efficiency, complementing the computational comparison of the two methods, would be to estimate the number of arithmetic operations required to advance a degree of freedom (DOF) one time unit. In this case such a comparison is not straightforward as the SBP-SAT-Lanczos method employs an adaptive and iterative procedure to evolve in time. Although the work associated with a single Lanczos iteration using a stencil width of $2p+1$ scale as $\sim 2(2p+1)$ per DOF it is not clear (from our experiments or from the literature) how to many iterations are needed to reach a certain tolerance. In general, the larger the timestep and grid spacing the fewer iterations are required to reach convergence. At the same time, the larger timesteps leads to worse accuracy in time. The time accuracy is hard to estimate and this makes a straight comparison with Hermite-Taylor difficult. Although a detailed analysis of the cost associated with the adaptive SBP-SAT-Lanczos may be possible it is beyond the scope of this paper.

3.3 Adaptive evolution of a free particle

To demonstrate the adaptive implementation we again simulate the evolution of the solution to (2.1) without a potential and with initial condition (3.3). Here we choose the $k_0=1$, corresponding to a wave that splits into a faster right-going wave and a slower left-going wave, see Fig. 7 where a time trace of the real part of the wave is plotted. We

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

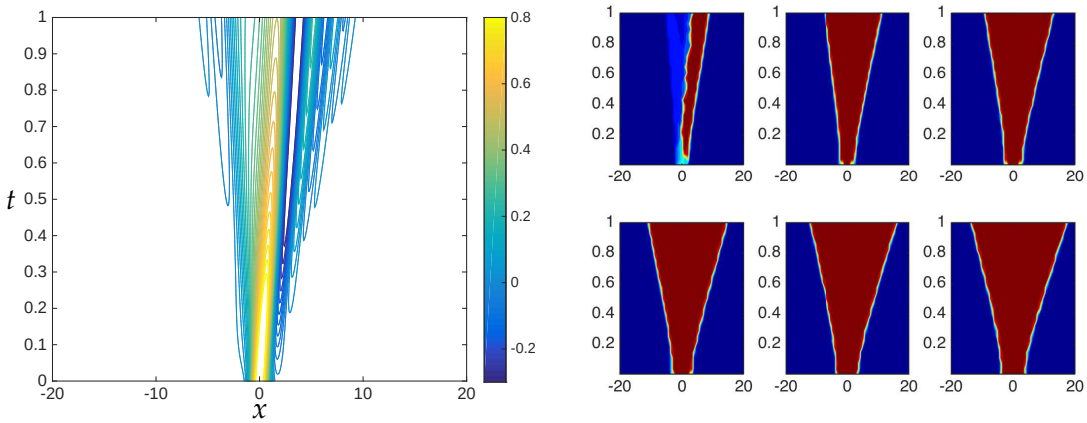


Figure 7: Adaptive simulation of a free particle. To the left, the real part of the approximate solution is plotted as a time trace. The initial data splits into two waves, the right-going moving fastest, and spreads over the computational domain as time passes. To the right the time traces of the change in the number, m, of derivatives kept at each node is plotted. From top left to bottom right the $\mathrm{TOL}=10^{-r}$, $r=2,\cdots,7$. The color scale goes from blue, $m=2$ to dark red $m=16$.

set $x_l=-20$ and $x_r=20$ so that there is no influence from the boundary conditions (which we take to be periodic for this computation). We set the $h=2/3$ and impose the upper limit on m to be 16. We also choose the time-step according to (3.1) with $C(m)=0.1/m$ and simulate up to time $t=1$. We perform the simulation for the tolerances, $\mathrm{TOL}=10^{-r}$, $r=2,\cdots,7$.

To the right in Fig. 7 the number of derivatives kept at each node, m, has been plotted as a function of space and time. As can be seen the order adapts itself according to the evolution of the solution. For the tolerances used the potential savings can be gauged by the amount of blue in the time traces to the right in Fig. 7. A conservative estimate is that the savings for this example could be a reduction of computational time to $1/3$ to $1/2$ of the computational time for a high order computation on a uniform grid.

To test if the criterion (2.23), used by the adaptive method, yields the requested tolerance we measure the error at $t=1$ and plot it for the different tolerances. The results can be found to the left in Fig. 8 and it can be seen that the error is of the same order of magnitude as the tolerance. The results are consistent for all the tolerances.

We also consider the conservation of the quantity $|1-\rho(t)/\rho(0)|$, where

$$\rho(t)=\int_{x_l}^{x_r} u^*(x,t)u(x,t)dx.$$

The results are plotted in the right part of Fig. 8 and show that the conservation properties appears to be very good but also that the conservation accuracy saturates as the tolerance is decreased.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

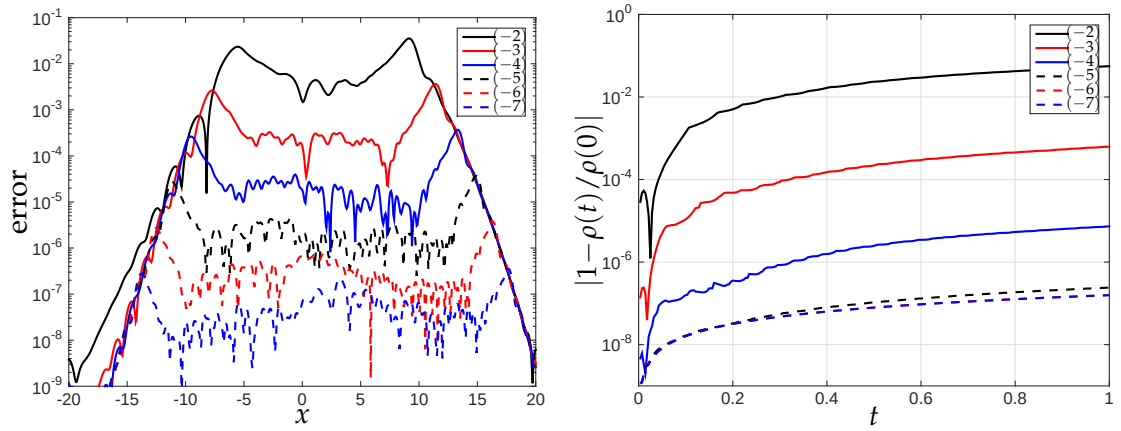


Figure 8: Errors for the adaptive method for the free particle problem. To the left we display the magnitude of the error as a function of x. To the right the quantity $|1-\rho(t)/\rho(0)|$, measuring the loss of conservation, is plotted as a function of time.

3.4 A Harmonic oscillator

We now proceed to solve a classic problem which includes a potential corresponding to a Harmonic oscillator. Here we choose the mass in such a way that the non dimensionalized equation takes the form

$$u_t = \frac{i}{2}u_{xx} - iV(x)u, \quad t>0, \quad x_l \le x \le x_r, \tag{3.9}$$

with the potential

$$V(x) = \frac{1}{4}x^2.$$

For the initial data

$$u(x,0) = \frac{e^{-x^2}}{\sqrt{\sqrt{\pi}}},$$

the real part of the exact solution is

$$\Re u(x,t) = u(x,0)\cos\frac{t}{2}.$$

We set $x_l = -8, x_r = 8$ and impose homogenous boundary conditions for the solution and the spatial derivatives. We discretize using $h = 16/8$ (i.e we use *nine* points) and solve until time $t = 5\pi$. With the potential present we found that the time step has to be taken a bit smaller for the method to remain stable, here we use $C(m) = 0.05/m$ to choose the time step.

For this computation we monitor the error in the real part of the solution. We also monitor the conservation of the quantity $|1-\rho(t)/\rho(0)|$ over time. The results for computations using $m = 2,\cdots,16$ can be found in Fig. 9 and in Table 3. As expected the higher

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

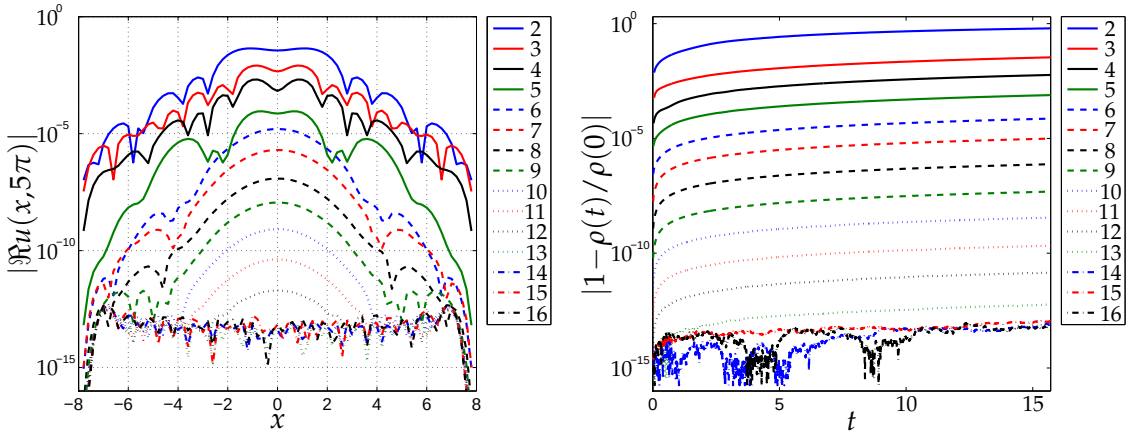


Figure 9: Errors for the non-adaptive method for the one dimensional harmonic oscillator problem. To the left we display the error in the real part of the solution. To the right the quantity $|1-\rho(t)/\rho(0)|$, measuring the loss of conservation, is plotted as a function of time.

Table 3: Maximum error in the real part of the solution at time $t=5\pi$ for various number of derivatives.

m		2	3	4	5	6	7	8
error		6.2(-2)	1.1(-1)	4.9(-2)	9.8(-3)	2.4(-3)	2.0(-4)	1.9(-5)
m	9	10	11	12	13	14	15	16
error	3.3(-6)	4.1(-7)	3.0(-8)	1.2(-9)	9.6(-12)	3.1(-12)	2.5(-12)	2.8(-12)

order methods completely outperform the lower order methods. It is also worth noting that the level of conservation is commensurate with the level of the error.

3.4.1 Simulation of a harmonic oscillator by the adaptive method

Our final example in one dimension uses the adaptive method to simulate (3.9) on $x\in[-20,20]$ with initial conditions consisting of the shifted Gaussian

$$u(x,0)=\frac{e^{-(x-1)^2}}{\sqrt{\sqrt{\pi}}}.$$

Here the real part of the solution satisfies

$$\Re u(x,t)=u(x,0)\cos\frac{t}{2},$$

at times $t=\pi,2\pi,3\pi,\cdots$, see Fig. 10.

For this example we set the maximum m to be 16, set $h_x=2/3$ and set the time step to be a third of the value used for the uniform method with $m=16$, $\Delta t=0.1/48h_x^2$. Here we solve the equation until time $t=3\pi$ and record the error at that time.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

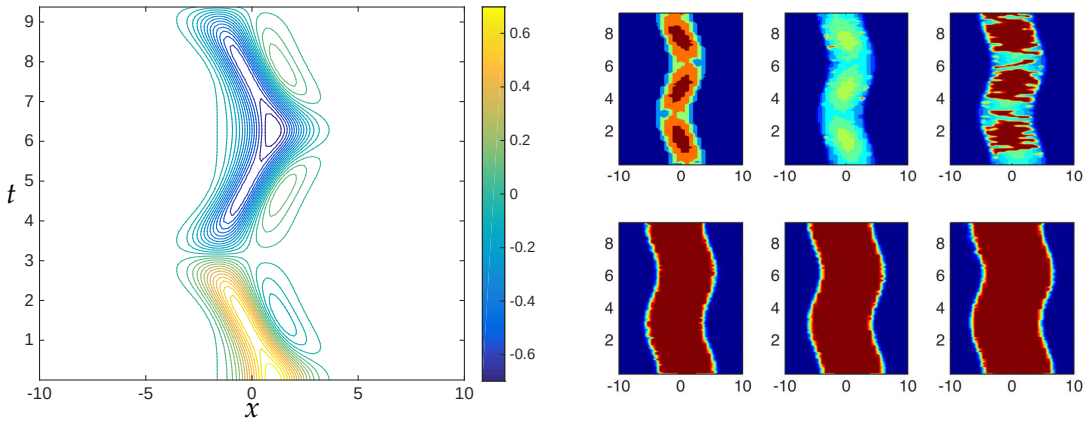


Figure 10: Adaptive simulation of a wave in a harmonic potential. To the left, the real part of the approximate solution is plotted as a time trace. The initial data oscillates back and forth inside the well as time passes. To the right the time traces of the change in the number, m, of derivatives kept at each node is plotted. From top left to bottom right the TOL$=10^{-r}$, $r=2,\cdots,7$. The color scale goes from blue, $m=2$ to dark red $m=16$.

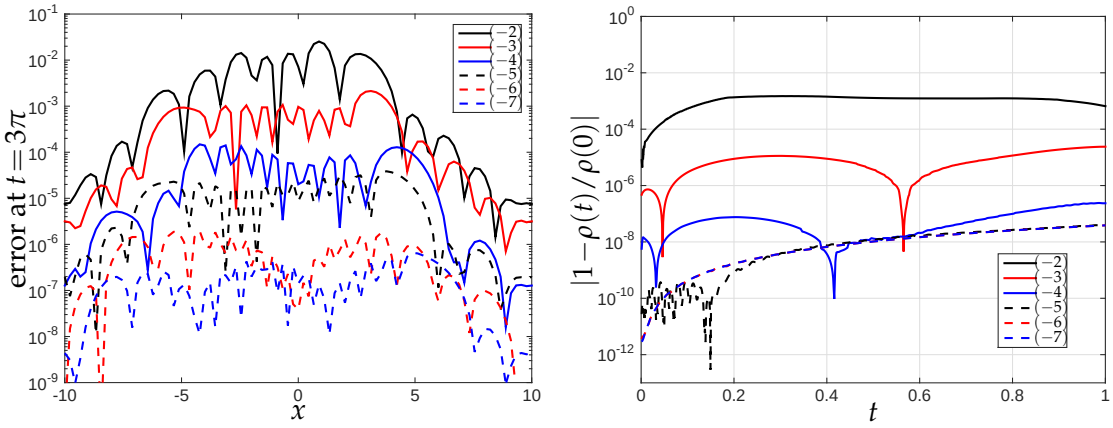


Figure 11: Errors for the adaptive method for the one dimensional harmonic oscillator problem. To the left we display the error in the real part of the solution. To the right the quantity $|1-\rho(t)/\rho(0)|$, measuring the loss of conservation, is plotted as a function of time.

As in the free particle example, we perform the simulation with tolerances TOL$=10^{-r}$, $r=2,\cdots,7$. We record the order as a function of time and as can be seen in the right part of Fig. 10 the adaptive method appears to work quite well with the order adapting itself according to the evolution of the solution. Also in this example the error levels at the end time are of the same order of magnitude as the prescribed tolerance. The conservation properties of the integral of the probability distribution is also similar as in the free particle example and appears to saturate at about 10^{-8}.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

3.5 A harmonic oscillator in two dimensions

As a final example we consider the Schrödinger equation for $u=u(x,y,t)$

$$\begin{aligned} &i\hbar\frac{\partial u}{\partial t}=-\frac{\hbar^2}{2M}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial x^2}\right)+V(x,y)u, \quad t>0, \quad (x,y)\in[x_l,x_r]\times[y_b,y_t], \\ &u(x,y,0)=u_0(x,y). \end{aligned} \tag{3.10}$$

Here we non-dimensionalize in a way equivalent to setting $M=1,\hbar=1$ in the above equation. We choose the potential to be the harmonic potential

$$V(x,y)=\frac{1}{2}\left(x^2+y^2\right).$$

With this potential it is easy to verify that equation (3.10) supports the solution

$$u(x,y,t)=Ae^{-it}e^{-\frac{(x^2+y^2)}{2}}. \tag{3.11}$$

Here we set $A=1/\sqrt{\sqrt{\pi}}$ and solve until $t=2\pi$ on the domain $(x,y)\in[-8,8]^2$. To set the time step we use $C(m)=0.05/m$.

Fig. 12 reports the errors as a function of $h_x=h_y$ for $m=2,\cdots,13$. As can be seen in the figure the slopes for $m=2,3,4,5,6$ are well defined and a least squares fits using the eight leftmost values yields the rates of convergence 3.9, 6.1, 8.1, 9.8 and 12.4, respectively. For higher m and errors down to $\sim 10^{-10}$ the methods become more accurate with increasing m but as the errors become even smaller the rates of convergence deteriorates and the error levels saturate at around $\sim 10^{-14}$. This is likely due to round-off effects.

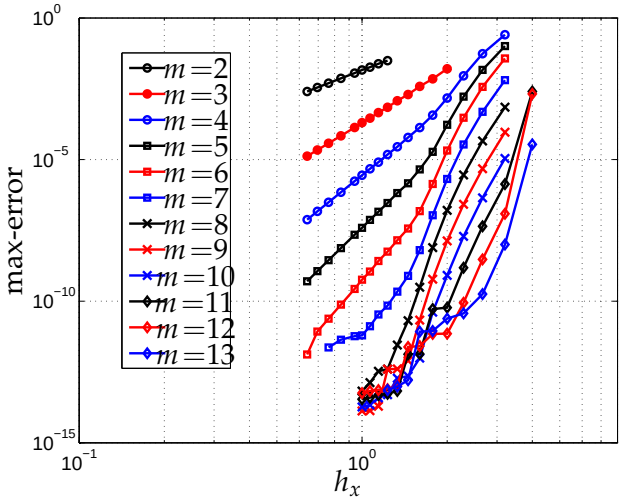


Figure 12: Max-error as a function of $h_x=h_y$ for $m=2,\cdots,13$.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

4 Summary

To summarize, we introduced an explicit and spectrally accurate (in time and space) method for solving the Schrödinger equation. We showed that the method can be used to accurately evolve solutions to Schrödinger equation on very coarse grids where the time step can be large. Problems with and without potentials in one and two dimensions were considered. We also introduced an order-adaptive method with a straightforward tolerance criterion and illustrated its efficiency by numerical examples.

Future extensions of the proposed method could include adaptive implementations in multiple dimensions and implementations with non-reflecting boundary conditions e.g. perfectly matched layers. The extension of the method to the non-linear Schrödinger equation with focusing and defocusing non-linearities could also be of interest.

Acknowledgments

Appelö was supported in part by NSF Grant DMS-1319054. Any conclusions or recommendations expressed in this paper are those of the author and do not necessarily reflect the views of the NSF.

References

[1] D. Appelö and T. Hagstrom. On advection by Hermite methods. *Paciffic Journal Of Applied Mathematics*, 4(2):125–139, 2011.
[2] Mark H. Carpenter, David Gottlieb, and Saul Abarbanel. Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: Methodology and application to high-order compact schemes. *J. Comput. Phys.*, 111(2):220 – 236, 1994.
[3] R. Chen and T. Hagstrom. *p*-adaptive Hermite methods for initial value problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46:545–557, 2012.
[4] X. Chen, D. Appelö, and T. Hagstrom. A hybrid Hermite–discontinuous Galerkin method for hyperbolic systems with application to Maxwell's equations. *Journal of Computational Physics*, 257, Part A:501 – 520, 2014.
[5] J. Goodrich, T. Hagstrom, and J. Lorenz. Hermite methods for hyperbolic initial-boundary value problems. *Math. Comp.*, 75:595–630, 2006.
[6] T. Hagstrom and D. Appelö. Solving PDEs with Hermite Interpolation. In *Springer Lecture Notes in Computational Science and Engineering*, 2015.
[7] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM J. Sci. Comput.*, 19:1552–1574, 1998.
[8] M. Hochbruck and A. Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 2010.
[9] K. Kormann and M. Kronbichler. Parallel finite element operator application: graph partitioning and coloring. *2011 Seventh IEEE International Conference on eScience*, pages 332–339, 2011.
[10] K. Kormann and E. Larsson. A Galerkin radial basis function method for the Schrödinger equation. *SIAM J. Sci. Comput.*, 35:A2832–A2855, 2013.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press

[11] H.-O. Kreiss and G. Scherer. Finite element and finite difference methods for hyperbolic partial differential equations. In *Mathematical Aspects of Finite Element in Partial Differential Equations*. Academic Press, Inc., 1974.
[12] K. Mattsson and J. Nordström. Summation by parts operators for finite difference approximations of second derivatives. *J. Comput. Phys.*, 199:503–540, 2004.
[13] Anna Nissen, Gunilla Kreiss, and Margot Gerritsen. High order stable finite difference methods for the Schrödinger equation. *J. Sci. Comput.*, 55:173–199, 2013.

https://doi.org/10.4208/cicp.080815.211116a Published online by Cambridge University Press