The EPS method: A new method for constructing pseudospectral derivative operators

Kristian Sandberg

Computational Solutions, LLC, 1800 30th St Suite 210B , Boulder, CO 80301

Keith J. Wojciechowski

University of Colorado Denver, 1250 14th St. Suite 600, P.O. Box 173364, Denver, CO 80217-3364

Abstract

We present a stable algorithm for constructing derivative matrices with small norm for pseudospectral methods. In particular, we construct second derivative matrices that incorporate Dirichlet or Neumann boundary conditions on an interval and on the disk. By construction, the norm of these matrices grows at the optimal rate $O(N^2)$ for N-by-N matrices, in contrast to standard pseudospectral constructions that result in $O(N^4)$ growth of the norm. The smaller norm has a big advantage when using the derivative matrix for solving time dependent problems such as wave propagation. The construction can be used with any quadrature, but we have found that by using quadratures based on Prolate Spheroidal Wave Functions, we can achieve a near optimal sampling rate close to 2 points per wavelength, even for non-periodic problems. We provide numerical results for the new construction and demonstrate that the construction achieves similar or better accuracy than traditional pseudospectral derivative matrices, while resulting in a norm that is orders of magnitude smaller than the standard construction. To demonstrate the advantage of the new construction, we also apply the method for solving the wave equation in constant and discontinuous media and for solving PDEs on the unit disk. We also present two compression algorithms for applying the derivative matrices in $O(N \log N)$ operations.

Email addresses: sandberg.kristian@gmail.com (Kristian Sandberg), keith.wojciechowski@email.ucdenver.edu (Keith J. Wojciechowski)

Keywords: Pseudospectral methods, Differentiation, Prolate Spheroidal Wave Functions, wave propagation, radial Laplacian

1. Numerical Comparisons of Methods

IN LEGENDS, REPLACE Nt WITH DT.

1.1. Single precision

1.1.1. Accuracy

Wave propagation of a sine mode. We first solve the following wave equation problem

$$\begin{array}{l} u_{tt} = u_{xx} + u_{yy}, \;\; (x,y) \in (-1,1) \times (-1,1) \\ u(x,\pm 1,t) = u(\pm 1,y,t) = 0 \\ u(x,y,0) = \sin(43\pi x)\sin(43\pi y) \\ u_t(x,y,0) = 0 \end{array} \quad . \tag{1}$$

The equation was discretized in space using three methods: the EPS method, and 8:th order finite difference scheme, and a standard pseudo-spectral derivative matrix with respect to Chebyshev-Lobatto nodes [REF]. To propagate the equation in time, we wrote the equation as a first order system in time of the form

$$\begin{bmatrix} u \\ v \end{bmatrix}_t = \begin{bmatrix} 0 & I \\ \Delta & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

where Δ denotes the Laplacian operator and I denotes the identity operator. We then used the Tal-Ezer method [REF] which effectively computes e^{tL} applied to a vector recursively by repeated application of L to a vector, where L is the operator

$$L = \begin{bmatrix} 0 & I \\ \Delta & 0 \end{bmatrix}.$$

We solved this equation in the interval $t \in [0,T]$ where T was chosen as the time corresponding to propagating the wavefield a distance of 100 spatial wavelengths. We define one spatial wavelength as the distance of one wavelength of our initial condition, which is $2/43$ in this case. For our first experiment, we adjusted the spatial and temporal sampling of the different method to all give roughly the same accuracy, which in our first experiment is roughly 4-5 digits. We measured the max error at each time step of the

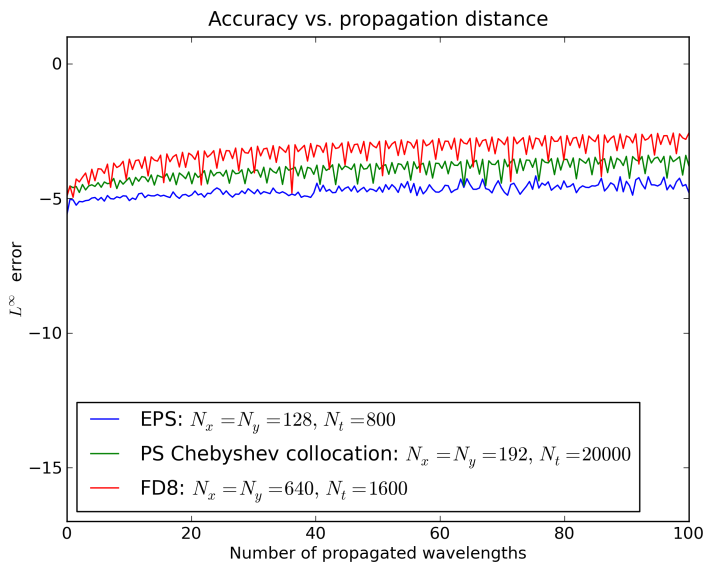

Figure 1: Accuracy of solving equation (1) in single precision using three different methods. $N_{x,y}$ and N_t denotes the number of spatial and temporal samples needed to achieve the accuracy.

propagation, and plot the error for the different methods as a function of propagated time in Figure 1. In Section 1.1.2 and Section 1.1.3 we will compare and analyze the memory requirements and computational costs for the methods.

Wave propagation of a pulse. As a second test example, we solve the 1D wave equation problem

$$\begin{array}{l} u_{tt} = u_{xx}, \ x \in (-1,1) \\ u(x,\pm 1,t) = 0 \\ u(x,y,0) = f(x) \\ u_t(x,y,0) = 0 \end{array}, \tag{2}$$

where the initial pulse $f(x) = e^{-1000x^2}$ (”Gaussian pulse”) or

$$f(x) = \begin{cases} \frac{1+\cos(\frac{\pi x}{0.05})}{2}, & |x| \leq 0.05 \\ 0, & |x| > 0.05 \end{cases}.$$

(”Cosine Bell”). The Gaussian pulse models a sharp, infinitely differentiable pulse, whereas the Cosine Bell models a sharp, but only a once differentiable pulse.

The equation was discretized in space using three methods: the EPS method, and 8:th order finite difference scheme, and a standard pseudo-spectral derivative matrix with respect to Chebyshev-Lobatto nodes [REF]. To propagate the equation in time, we used the Tal-Ezer method as described in the previous example.

We solved this equation in the interval $t \in [0, T]$ where T was chosen as 400 seconds, which corresponds to the time it takes to propagate the pulse across the interval back and forth 100 times. We plot the error for the different methods as a function of propagated time in Figure 2. In Section

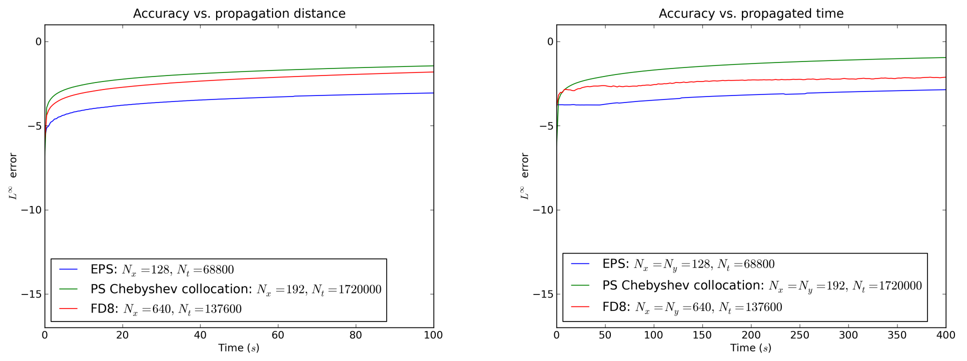

Figure 2: Accuracy of solving equation (2) with a Gaussian pulse (left) and Cosine Bell (right) in single precision using three different methods. $N_{x,y}$ and N_t denotes the number of spatial and temporal samples needed to achieve the accuracy.

1.1.2 and Section 1.1.3 we will compare and analyze the memory requirements and computational costs for the methods.

1.1.2. Spatial and temporal sampling

In this section we analyze and compare the spatial sampling and the time steps needed to achieve the accuracy in Section 1.1.1. By trying to reach similar accuracies for all three methods, the spatial sampling will give us an estimate how much memory is needed to achieve a given accuracies. To compare the necessary memory requirements, we computed the memory to store one wavefield $u(\mathbf{x})$ in one, two, and three dimensions. Note that depending

	1D	2D	3D
EPS	512 B	262 kB	134MB
PS (standard)	768 B	590 kB	453 MB
FD8	2560 B	6.55 MB	16.8 GB

Table 1: Memory requirement (in Bytes) per wave field (single precision).

	1D	2D	3D
EPS	1	1	1
PS (standard)	1.5	2.25	3.38
FD8	5	25	125

Table 2: Relative memory requirement per wave field (single precision)

on the time stepping scheme being used, one needs to store several such wave fields. In our experiments, we used the same time stepping methods for all methods, which means that the net memory requirements are proportional to the memory requirement for storing one wave field.

In Table 1 we list the memory requirement to achieve the accuracy for the type of examples in Section 1.1.1. The memory requirements were computed by the formula $4N_x^d$, where the factor 4 comes from considering single precision, and d denotes the dimensionality. The number of spatial samples N_x for the above experiments were 128, 192, and 640 for the EPS method, the standard PS method, and 8:th order finite difference scheme, respectively. In Table 2 we have tabulated the relative memory requirements for easy comparisons of the methods. In Table 3 we list the relative time step requirement to achieve the accuracy for the type of examples in Section 1.1.1.

EPS	1
FD8	0.5
PS (standard)	0.04

Table 3: Relative time step (single precision)

1.1.3. Computational cost

Although the finite difference scheme requires more memory than the other methods, a finite difference application is typically considerably faster than the other method when comparing identical sizes. In order to obtain a fair comparison between the methods, we will in this section compare the computational speeds between the methods by first measuring the CPU time for evaluating the second derivative for the different methods for sampling rates that gives roughly the same accuracy.

The timings were performed as follows: For the EPS and the PS method, we measured the CPU time for applying the second derivative operator as dense matrix-vector applications. It should be noted that both of these methods may benefit from using any of the fast application methods discussed in Section **??**. However, speed gains from such fast methods are highly dependent on computer architecture and implementation. In general, such fast application methods only win over dense matrix-vector multiplication for relatively large sizes.

For the 8:th order finite difference method we measured the CPU time for applying the second derivative using a 9-tap finite difference stencil, using a cache-friendly implementation. All tests were done on an iMac with a 2.8 GHz Intel Core i5 Nehalem processor with 8 MB Level 3 cache and 4 GB of 1333 MHz DDR3 RAM. The code was implemented in Fortran 2003 and compiled with Intel's Fortran compiler XE 12.0 for OS X, using the MKL library routines for the matrix-vector multiplications. In cases where the CPU time was too short to give reliable timing estimates from a single application of the spatial operator, the result was averaged over as many applications that were needed to reliably measure the CPU time. All timings were done single threaded.

In Table 4 we have listed the CPU time measured for each method. For 2D and 3D data, one has to apply the spatial operators on both data that is aligned in memory, and data that is not aligned in memory. We therefore measure the time for these cases seprately in Table 4.

Using our timings for applying the spatial operator, we now compute the total computational time according to the following formula:

$$\begin{aligned}
T_{total} &= N_t N_{it} T_{align}(N_x), \quad \text{(1D)} \\
T_{total} &= N_t N_{it} (T_{align}(N_x) N_y + T_{no-align}(N_y) N_x), \quad \text{(2D)} \\
T_{total} &= N_t N_{it} (T_{align}(N_x) N_y N_z + T_{no-align}(N_y) N_x N_z + T_{no-align}(N_z) N_x N_y), \ \text{(3D)}
\end{aligned} \tag{3}$$

Method	Aligned data	Not-aligned data
EPS (dense)	2.6E-06 s	3.2E-06 s
PS (standard, dense application)	5.2E-06 s	6.3E-06 s
PS (standard, FFT application)	8.7E-07 s	1.7E-06 s
FD8	4.7E-06 s	7.4E-06

Table 4: Cost for one spatial operator application (single precision) to achieve similar accuracy for three different methods.

Method	1D	2D	3D
EPS	0.021 s	5.9 s	1200 s
PS (standard, dense)	1.0 s	440 s	1.3E+05 s
PS (standard, FFT)	0.17 s	99 s	3.1E+04 s
FD8	0.075 s	120	1.3E+05 s

Table 5: Total computational cost to achieve 4-5 digits accuracy for the type of problem in Section 1.1.1.

where N_t denotes the total number of time steps, $N_{x,y,z}$ the number of samples in the x-,y-, and z-directions, N_{it} denotes the number iterations used for the Tal-Ezer recursion, $T_{align}(N)$ the CPU time for applying a spatial second derivative on N data samples aligned in memory, and $T_{no-align}(N)$ the CPU time for applying a spatial second derivative on N data samples not aligned in memory.

In our case, we will always use $N_x = N_y = N_z$. For all method, we used $N_{it} = 10$. Applying this timing formula using the required sampling to achieve the accuracy in Section 1.1.1, we list the total CPU time in Table 5.

1.2. Double precision

1.2.1. Accuracy

In this section we repeat the experiments from Section 1.1.1, but now target a higher accuracy using double precision computations.

Wave propagation of a sine mode. In Figure 3 we plot the accuracy from solving (1). Although we tried to achieve the same accuracy for all three methods by increasing the sampling, the 8:th order finite difference scheme is unable to achieve as high accuracy as the spectral methods. This, we believe, is mainly due to numerical dispersion, which even high order finite difference schemes handle poorly (See [INSERT REF TO FORNBERG GRAPHS] for an illustration of this). Even the standard pseudo-spectral method is unable to achieve the high accuracy of the EPS method. REDO THIS ONE

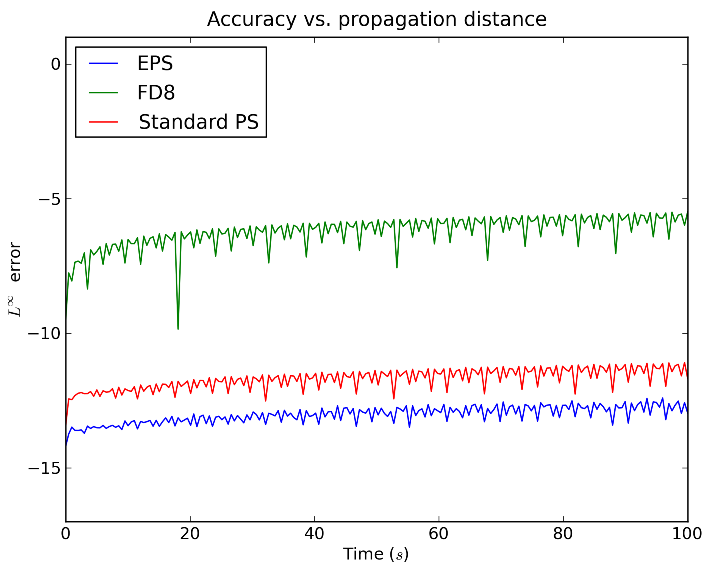

Figure 3: Accuracy of solving equation (1) in double precision using three different methods. $N_{x,y}$ and N_t denotes the number of spatial and temporal samples needed to achieve the accuracy.

BUT USE 256 (OR MORE) NODES INSTEAD. ADD SAMPLING TO THE FIGURE LEGEND.

Wave propagation of a pulse. In Figure 4 we plot the accuracy from solving (2) targeting 12-13 digits accuracy using double precision.

1.2.2. Spatial and temporal sampling

In Table 6 we list the memory requirement to achieve the accuracy for the type of examples in Section 1.2.1. The memory requirements were computed by the formula $8N_x^d$, where the factor 8 comes from considering double precision, and d denotes the dimensionality. The number of spatial samples N_x

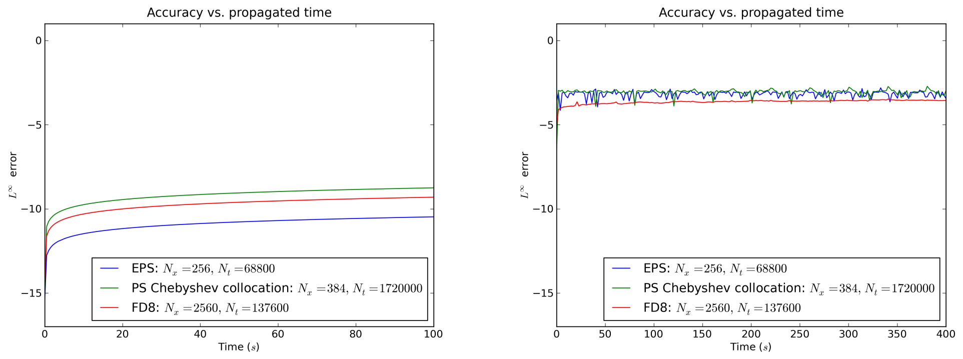

Figure 4: Accuracy of solving equation (2) with a Gaussian pulse (left) and Cosine Bell (right) in double precision using three different methods. $N_{x,y}$ and N_t denotes the number of spatial and temporal samples needed to achieve the accuracy.

	1D	2D	3D
EPS	2.0 kB	4.1 MB	8.6 GB
FD8	20 kB	420 MB	8.6 TB
PS (standard)	3.1 KB	9.4 MB	29 GB

Table 6: Memory requirement per wave field (double precision)

for the above experiments were 256, 384, and 2560 for the EPS method, the standard PS method, and 8:th order finite difference scheme, respectively. In Table 8 we list the relative time step requirement to achieve the accuracy for the type of examples in Section 1.2.1.

1.2.3. Computational cost

In this section we use the same methodology as in Section 1.1.3 to compare the computational times for the different methods. We first list the CPU time in Table 9 for applying the spatial operator in one dimension for data aligned and not aligned in memory using the sampling rates 256, 384, and 2560 for the EPS method, the standard PS method, and 8:th order finite difference scheme, respectively. We next use (3) to derive the computational times for the different methods. For all method, we used $N_{it} = 16$. Applying this timing formula using the required sampling to achieve the accuracy in Section 1.2.1, we list the total CPU time in Table 10.

	1D	2D	3D
EPS	1	1	1
FD8	10	100	1000
PS (standard)	1.5	2.3	3.4

Table 7: Relative memory requirement per wave field (double precision)

EPS	1
FD8	0.125
PS (standard)	0.02

Table 8: Relative time step (double precision)

Method	Aligned data	Not-aligned data
EPS	2.0E-05 s	2.3E-05 s
PS (standard, dense)	4.5E-05 s	5.2E-05 s
PS (standard, FFT)	2.9E-06 s	6.5E-06 s
FD8	3.1E-05 s	1.1E-04

Table 9: Cost for one spatial operator application (double precision) to achieve similar accuracy for three different methods..

Method	1D	2D	3D
EPS	0.26 s	141 s	5.5E+04 s
PS (standard, dense)	29 s	2.4E+04 s	1.4E+07 s
PS (standard, FFT)	1.9 s	2.3E+03 s	1.5E+06 s
FD8	3.2 s	3.7E+04 s	N/A

Table 10: Total computational cost to achieve 12-13 digits accuracy for the type of problem in Section 1.2.1. Note that we do not list the cost for computing the 3D solution for the 8:th order finite difference scheme, as this would require more RAM than is currently available on a single machine.

1.3. Qualitative comparison in discontiuous media

We finally show compare the EPS method for wave propagation in discontiuous media by solving the equation

$$\begin{aligned} &u_{tt} = c(x,y)\Delta u,\ x \in (-1,1) \\ &u(x,\pm 1,t) = u(\pm 1,y,t) = 0 \\ &u(x,y,0) = e^{-1000\left((x+\frac{1}{2})^2+(y+\frac{1}{2})^2\right)}, \\ &u_t(x,y,0) = 0 \end{aligned} \tag{4}$$

where the velocity is given by

$$c(x,y) = \begin{cases} 2, & (x,y) \in [0,0.5]\times[0,0.5] \\ 1, & (x,y) \notin [0,0.5]\times[0,0.5] \end{cases}$$

which models a sharp pulse centered initially in the upper right corner, and propagating through a domain with a square region in the lower left corner with a higher velocity. We solve this equation using the method from Section 1.1.1.

We show the resulting wave field using the EPS method with 128 APSWF nodes in each dimension after $t = 0.84s$ in Figure 6. In Figure **??** we compare a slice of the wavefield with that generated by the PS method using 128 Chebyshev nodes in each dimension, and a time step 1/25:th of the time step used by the EPS method. The slice was chosen along the diagonal from the upper left to the lower right, and cuts accross one of the corner discontinuities at the center of the domain. We notice that despite the significantly larger time step used by the EPS method, the EPS method produces a significantly cleaner result.

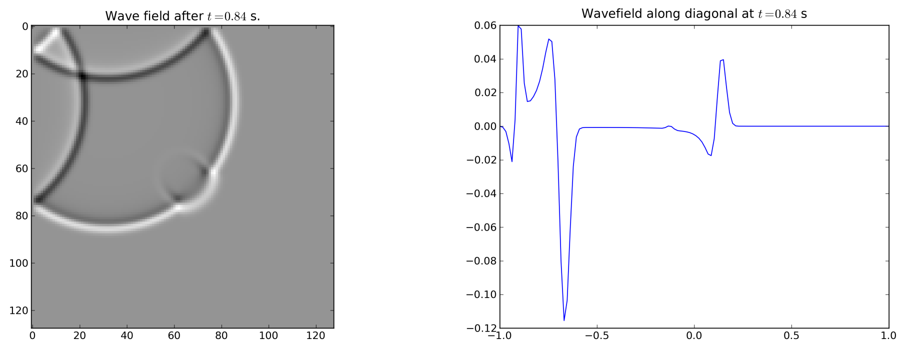

Figure 5: The resulting wave field after $t = 0.84$ s from solving Equation (4) using the EPS method with 128 APSWF nodes in each dimension.

.

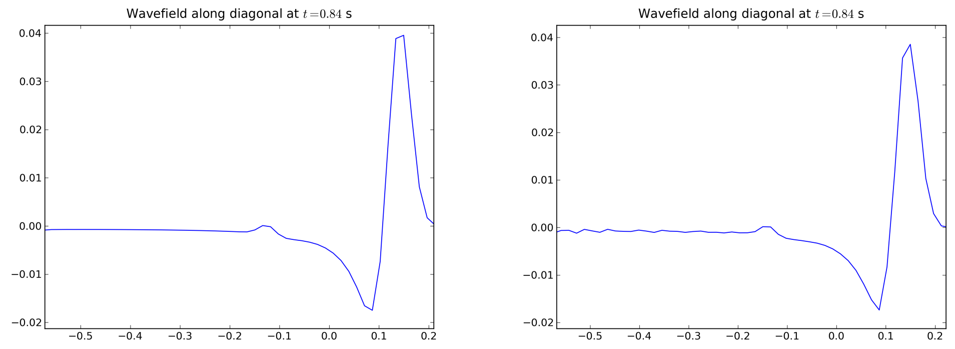

Figure 6: Comparison of the resulting wave field after $t = 0.84$ s from solving Equation (4) using the EPS method with 128 APSWF nodes in each dimension (left) and using the PS method with 128 Chebyshev nodes in each dimension (right).

.