Sparse Grid Discretizations based on a Discontinuous Galerkin Method

Alexander B. Atanasov[1,2] and Erik Schnetter[1,3,4]

[1]*Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada*
[2]*Department of Physics, Yale University, New Haven, CT, USA*
[3]*Department of Physics, University of Guelph, Guelph, ON, Canada*
[4]*Center for Computation & Technology, Louisiana State University, LA, USA*

Abstract.

I. INTRODUCTION

It is very costly to accurately discretize functions living in four or more dimensions, and even more expensive to solve partial differential equations (PDE) in such spaces. The reason is clear – if one uses N grid points or basis functions for a one-dimensional discretization, then a straightforward tensor product ansatz leads to $P = N^D$ total grid points or basis functions in D dimensions. For example, doubling the resolution N in the time evolution of a three-dimensional system of PDEs increases the cost by $2^4 = 16$. As a result, such calculations usually require large supercomputers, often run for weeks and months, and one still cannot always reach required accuracies.

Higher-dimensional calculations up to $D = 7$ might be needed to discretize phase space e.g. to model radiation transport in astrophysical scenarios. These are currently only possible in an exploratory manner, i.e. while looking for qualitative results instead of achieving convergence. This unfortunate exponential scaling with the number of problem dimensions is known as the *Curse of Dimensionality* [1].

Sparse grids offer a conceptually elegant alternative. They were originally constructed in 1963 by Smolyak [2], and efficient computational algorithms were then developed in the 1990s by Bungartz, Griebel, Zenger, Zumbusch, and many collaborators. Sparse grids employ a discretization that is not based on tensor products, and can in the best case offer costs that grow only linearly (sic!) or log-linearly with the linear resolution N. At the same time, the accuracy is only slightly reduced by a factor logarithmic in N. See [3, 4] for very readable introductions and pointers to further literature. Since these introductions are very well written we will not repeat them here. Familiarity with these introductions might make this paper easier to read.

Sparse grid algorithms are very well developed. They exist e.g. for arbitrary dimensions, higher-order discretizations, and adaptive mesh refinement, and have been shown to work for various example PDEs, including wave-type and transport-type problems. The respective grid structure has a very characteristic look (it is sparse!) and is fractal in nature. See figure 1 for an example.

Surprisingly, sparse grids are not widely known in the physics community, and are not used in astrophysics or numerical relativity. We conjecture that this is largely due to the non-negligible complexity of the recursive D-dimensional algorithms, and likely also due to the lack of efficient open-source software libraries that allow experimenting with sparse grids. Here, we set out to resolve this:

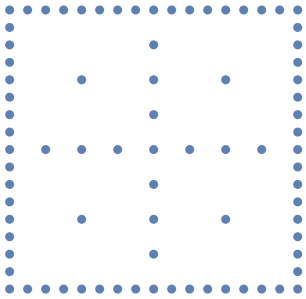

FIG. 1: The collocation points for a particular sparse grid with 4 levels in two dimensions. Note the typical structure which is dense along the edges and fractally sparse in the interior.

1. We develop a novel sparse grid discretization method based on Discontinuous Galerkin Finite Elements (DGFE), based on work by Guo and Wang [5, 6]. Most existing sparse grid literature is instead based on finite differencing. DGFE methods are more local, which leads to superior computational efficiency on today's CPUs.

2. We solve the scalar wave equation in $3+1$ and $5+1$ dimensions, comparing resolution, accuracy, and cost to standard ("full grid") discretizations. We demonstrate that sparse grids are indeed vastly more efficient.

3. We introduce a efficient open source library for DGFE sparse grids. This library is implemented in the Julia programming language [7]. All examples and figures in this paper can be verified and modified by the reader.

II. CLASSICAL SPARSE GRIDS

The computational advantage of sparse grids comes from a clever choice of basis functions, which only include important degrees of freedom while ignoring less important ones. The meaning of "important" in the previous sentence is made concise via particular function norms, as described in [3, 4] and references therein. We will omit the motivation and proofs for error bounds here, and will only describe the particular basis functions used in sparse grids.

Discretizing a function space V means choosing a particular set of basis functions $\phi_{l,i}$ for that space. We assume that the basis functions are ordered by a *level* ℓ, and each level contains a set of *modes*, here indexed by i. This discretization allows *approximating* a function by only considering levels $0 \le \ell \le L$, i.e. by dropping all basis function with level $\ell > L$. For example, the real Fourier basis for analytic functions with 2π periodicity consists of $\phi_{k,0}(x) = \cos kx$ and $\phi_{k,1}(x) = \sin kx$. Here, each level k (except $k = 0$) has 2 modes. The notion of a level is important if one changes the accuracy of the approximation: one always includes or excludes all modes within a level. We will need the notion of levels and modes (or *locations*) below. **TODO: ES: should we keep the notion of a "location", or can we remove it? Definitely lets keep it, because in the DG case we have the "mode" corresponding to the degree of the polynomial and the "location"/"element" corresponding to the multi-interval that its supported in**

A finite differencing approximation can be defined via a nodal basis. For example, a second-order accurate approximation where the discretization error scales as $O(N^{-2})$ for N

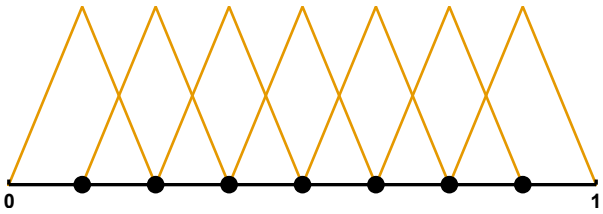

FIG. 2: Nodal basis of hat function with $N = 7$ basis functions.

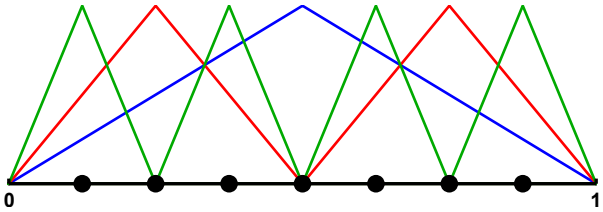

FIG. 3: A hierarchical basis for the same space as before, using levels $\ell = 0, 1, 2$.

basis functions (grid points) uses triangular (hat) functions, as shown in Figure 2, for a piecewise linear continuous ansatz. Level ℓ has 2^ℓ basis functions, so that a discretization with cut-off L has $N = 2^{(L+1)} - 1$ basis functions in total.

The examples given here in the introduction assume homogenous boundaries for simplicity, i.e. they assume that the function value at the boundary is 0. This restriction can easily be lifted by adding two additional basis functions, as we note in section II A below.

The basis functions for sparse grids in multiple dimensions are defined via a one-dimensional *hierarchical basis*. This hierarchical basis is equivalent to the nodal basis above, in the sense that they span the same space when using the same number of levels L, and one can efficiently convert between a them with cost $O(N \log N)$ without loss of information. Figure 3 shows such a hierarchical basis with 3 levels.

A full set of nodal basis functions in multiple dimensions, corresponding to a grid as used in a finite differencing approximation, can be constructed as tensor product of one-dimensional nodal basis functions. A full set of hierarchical basis functions is likewise constructed as a tensor product.

A. The Hierarchical Sparse Basis

For an in-depth exposition on sparse grids, see [3, 4]. We will follow the conventions of the former. Historically, sparse grids were constructed from a basis of "hat" functions, defined by taking appropriate shifts and scalings of an initial hat function ϕ:

$$\phi(x) := \begin{cases} 1 - |x|, & \text{if } x \in [1, -1] \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

$$\phi_{n,i}(x) = \phi\left(2^n \cdot \left(x - i \cdot 2^{-n}\right)\right). \tag{2}$$

to obtain a basis of functions supported on the unit interval $[0, 1]$. For a fixed n, we denote this space in terms of the nodal basis by

$$\mathbf{V}_n := \text{span}\left\{\phi_{n,i},\ 1 \le i \le 2^n - 1\right\}. \tag{3}$$

Here, n is a positive integer, termed the *level* of resolution, whose size determines the fineness of the approximation. Each element $\phi_{n,i}$ is localized to within a radius 2^{-n} neighborhood around the point $i \cdot 2^{-n}$. Here we use the variable i, ranging from 1 to 2^n, to refer to the respective *location* at level n. **TODO: ES: earlier we introduced "levels" and "modes". now we introduce elements. shouldn't we use the term "mode" (or maybe "location") instead of "element" here? then we can introduce "element" only later when discussing DGFE methods. Fixed to location**

For a function $u(x)$ on $[0, 1]$, we can build its representation $\tilde{u}_n$ in this basis by

$$\tilde{u}_n(x) = \sum_{i=1}^{2^\ell - 1} u_{n,i}\, \phi_{n,i}(x). \tag{4}$$

Here $u_{n,i}$ are the *coefficients* of u in this basis representation on $\mathbf{V}_n$.

In fact, $\mathbf{V}_n \subseteq \mathbf{V}_{n+1}$ and we can pick a decomposition

$$\mathbf{V}_n = \bigoplus_{\ell \le n} \mathbf{W}_\ell. \tag{5}$$

so that the subspace $\mathbf{W}_\ell$ is complementary to $\mathbf{V}_{\ell-1}$ in $\mathbf{V}_\ell$, and $\mathbf{W}_1 = \mathbf{V}_1$. $\mathbf{W}_\ell$ can in fact be defined explicitly as

$$\mathbf{W}_\ell := \text{span}\left\{\phi_{\ell,i} : 1 \le i \le 2^\ell - 1,\ i \text{ odd}\right\}. \tag{6}$$

When accounting for nonvanishing boundary conditions, we add in a subspace, sometimes denoted by $\mathbf{W}_0$ and defined to be span $\{x, 1 - x\}$. For the sake of simplicity, we will ignore this subspace in this section. **TODO: ES: i added a similar comment above; please point to the section where this subspace is introduced. This is here. It isn't mentioned again anywhere else**

A function u can now be represented on $\mathbf{V}_n$ in terms of the $\phi_{\ell,i}$ basis as

$$\tilde{u}_n(x) = \sum_{\ell=0}^{n} \sum_{\substack{i=1 \\ i \text{ odd}}}^{2^\ell - 1} u_{\ell,i}\, \phi_{\ell,i}(x) \tag{7}$$

where again here $u_{\ell,i}$ are the coefficients of $u(x)$ in the hierarchical representation.

This new choice of basis functions, $\phi_{\ell,i}$ with ℓ ranging from 1 to n and i odd is called the *hierarchical* or *multi-resolution* basis for $\mathbf{V}_n$, because it includes terms from each resolution level. This basis is especially convenient when one wants to increase the resolution (i.e. the number of levels n), as one does not have to recalculate existing coefficients: Adding new levels with all coefficients set to 0 does not change the represented function.

In higher dimensions, the basis functions are obtained through the tensor product construction [3]. Now, the multi-resolution basis for D dimensions consists of a set of basis functions $\phi_{\mathbf{l},\mathbf{i}}$ indexed by a D-tuple, $\mathbf{l}$, of levels, called a *multi-level*, and a second tuple of

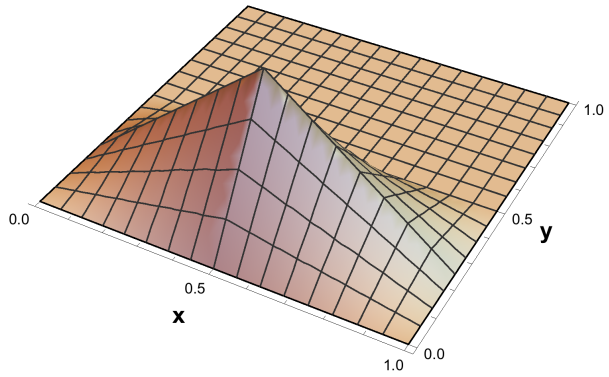

FIG. 4: A tensor product of two hat functions, namely the one at level $\ell = 0$ along x and one of the two at level $\ell = 1,\ i = 1$ along y. This corresponds to $\phi_{\mathbf{l}=(1,2),\,\mathbf{i}=(1,1)}$

elements $\mathbf{i}$, called a *multi-element*. The tensor product construction defines

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \prod_{d=1}^{D} \phi_{\mathbf{l}_d,\mathbf{i}_d}(\mathbf{x}_d), \tag{8}$$

$$\mathbf{W}_\mathbf{l} := \bigotimes_{d=1}^{D} \mathbf{W}_{\mathbf{l}_d} \tag{9}$$

$$\Rightarrow \mathbf{W}_\mathbf{l} = \text{span}\left\{\phi_{\mathbf{l},\mathbf{i}} : 1 \le \mathbf{i}_j \le 2^{\mathbf{l}_j},\ \mathbf{i}_d \text{ odd for all } d\right\} \tag{10}$$

Figure 4 shows the two-dimensional basis function $\phi_{\mathbf{l}=(1,2),\mathbf{i}=(1,1)}$ as example. We write $\mathbf{k} \le \mathbf{l}$ if the inequality holds for all components, i.e. $\forall d\ \mathbf{k}_d \le \mathbf{l}_d$. Then,

$$\mathbf{V}_\mathbf{l} := \bigotimes_{d=1}^{D} \mathbf{V}_{\mathbf{l}_j} = \bigoplus_{\mathbf{k}\le\mathbf{l}} \mathbf{W}_\mathbf{k} \tag{11}$$

TODO: ES: add a comment regarding the "product" and "sum" spaces, to help people understand the notation, and how the basis functions for the vector spaces are constructed via tensor products. Not sure what more to add to help in understanding

Often in numerical approximation, the maximum level along each axis is the same number n, i.e. $\mathbf{l} = (n, n, \ldots, n)$. In this case we write

$$\mathbf{V}_n^D := \mathbf{V}_{(n,n,\ldots,n)} = \bigoplus_{|\mathbf{k}|_\infty \le n} \mathbf{W}_\mathbf{k} \tag{12}$$

We then define the sparse subspace $\hat{\mathbf{V}}_n^D \subseteq \mathbf{V}_n^D$ in terms of the hierarchical subspaces $\mathbf{W}_\mathbf{k}$.

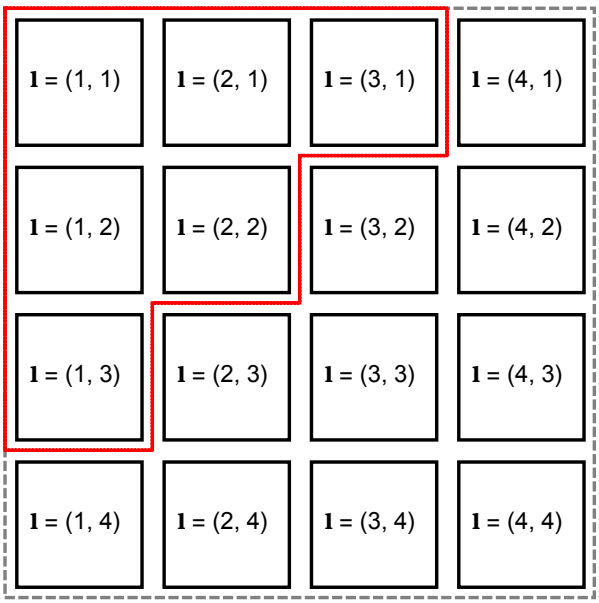

FIG. 5: Illustration of the "sparsification" of a set of full grid coefficients for $D = 2, n = 4$ by removing all multi-levels past a certain 1-norm. The dashed outline denotes the original multi-levels included in the full space, while the red line denotes only those included in the sparse space. Note that levels with higher indices contain exponentially more basis functions, so that the resulting reduction in dimensionality is significant, from $O(N^D)$ to $O(N \log^{D-1} N)$, as elucidated in the main text in section II B.

Rather than taking a direct sum over all the multi-levels $\mathbf{k}$, we define [1]

$$\hat{\mathbf{V}}_n^D := \bigoplus_{|\mathbf{k}|_1 \leq n} \mathbf{W}_\mathbf{k} \tag{13}$$

to be the sparse space. **TODO: ES: why include the number of dimensions d here, and not introduce this notation earlier? it would be less confusing if we said $\mathbf{V}_\ell^D$ earlier as well instead of $\mathbf{V}_\mathbf{l}$. The only reason being that the tuple l holds the information about the dimension by virtue of being a D-tuple** Note that we use here the 1-norm instead of the ∞-norm to decide which multi-level basis indices to include. This selects only $1/d!$ of the possible multi-levels. This subspace selection is illustrated in Figure 5 for $d = 2, n = 3$.

Because different levels have a different numbers of basis functions, and this process cuts off higher levels that have exponentially more basis functions than the lower ones, the resulting dimensionality reduction is significant. We discuss this in detail in the next subsection.

[1] Note this is a slightly definition from that used in [3], namely $|\mathbf{k}|_1 \leq n + D - 1$. Our definition makes the direct sum considerably cleaner and translates more nicely to the DG case later.

B. Comparing Costs

To construct a representation of a function living in $\mathbf{V}_n^D$, 2^n basis coefficients must be determined along each axis. This number is often referred to as the number of collocation points or grid points along that dimension, and we denote it by N.

The dimension of $\mathbf{V}_n^D$ is the number of coefficients that must be stored for a representation up to level n **TODO: ES: use ℓ to denote a level instead of n Since ℓ was used for individual level, I suggest we stick with n or use L** along each axis, and it is straightforward to see that we can write this as

$$\dim \mathbf{V}_n^D = O(2^{nD}) = O(N^D). \tag{14}$$

This exponential dependence of the size of this space on the number of spatial dimensions D is the manifestation of the curse of dimensionality. On the other hand, the dimension of the sparse space is

$$\dim \hat{\mathbf{V}}_n^D = \sum_{|\mathbf{k}|_1 \leq n} 2^{|\mathbf{k}|_1} = O(2^n n^{D-1}) = O(N \log^{D-1} N). \tag{15}$$

The sparse space consists of far fewer coefficients, and suppresses the exponential dependence to only the logarithmic term $\log N$.

Moreover, [3, 8] show that the two-norm of the error of an approximation $\tilde{u}_{\text{full}} \in \mathbf{V}_n^D$ of a function $u \in H_2([0,1]^D)$ in the Sobolev space **TODO: ES: which Sobolev space?I state $H_2([0,1]^D)$, is there more info to include?** scales as

$$||u(\mathbf{x}) - \tilde{u}_{\text{full}}(\mathbf{x})||_2 = O(N^{-2}) \tag{16}$$

while for $\tilde{u}_{\text{sparse}}$ in the corresponding sparse space, it is

$$||u(\mathbf{x}) - \tilde{u}_{\text{sparse}}(\mathbf{x})||_2 = O(N^{-2} \log^{D-1} N). \tag{17}$$

TODO: ES: introduce notations $\tilde{u}$, $\hat{u}$ somewhere earlier I'll introduce the convention that $\tilde{u}$ means discrete representation of u, and forget about using the $\hat{u}$. This means that the error increases only by a logarithmic factor, showing the significant advantage of the sparse basis over the full basis for all dimensions greater than one.

We define the *ε-complexity* [3, 9] of a scheme as the number of coefficients P required to reach error ϵ. A full grid has an epsilon complexity in the L_2 norm of

$$\varepsilon_{L_2}^{\text{full}}(P) = O(P^{-2/D}) \tag{18}$$

which decreases quite slowly for larger dimensions. The sparse grid, on the other hand, does considerably better:

$$\varepsilon_{L_2}^{\text{sparse}}(P) = O(P^{-2} \log^{3(D-1)} P)). \tag{19}$$

For higher-order discretizations, the power P^{-2} increases to e.g. P^{-4} or P^{-8} etc. for both full and sparse grids. **TODO: AA: I don't think this last sentence makes sense before we've introduced the higher order DG scheme**

Though the logarithmic factor is significant enough that it cannot be ignored in practice, even in high dimensions, this still represents a significant reduction in the number of coefficients required.

III. THE DISCONTINUOUS GALERKIN SPARSE BASIS

We review here the work of [5] and [6] on how one can generalize this sparse grid construction beyond hat functions or their higher-order finite-differencing equivalents. (Although we only discussed second-order accurate sparse grids in the previous section, these can be extended to arbitrary orders in a straightforward manner.) Then we introduce a scheme for defining a derivative operator in a manner similar to the Operator-Based Local Discontinuous Galerkin (DG) Method developed in [10].

The main advantage of DGFE sparse grids over classical, finite-differencing based sparse grids is the same as the advantage that regular DGFE discretizations posess over finite-difference discretization: (1) for high polynomial orders (say, $k > 8$), the collocation points within DGFE elements can be clustered near the element boundaries, leading to increased stability and accuracy near domain boundaries, and (2) the derivative operators for DGFE discretizations exhibit a significant block structure, which is computationally more efficient than the band structure found in higher-order finite differencing operators.

A. Constructing Galerkin Bases

The space $\mathbf{V}_n$ of hat-functions in one dimension is equivalent to the span of 2^n hat functions defined on sub-intervals of length 2^{-n} that subdivide $[0,1]$ into 2^n intervals of the form $I_{i,n} := [(i-1)\cdot 2^{-n},\, i\cdot 2^{-n}]$. A natural generalization of this is to consider the linear sum of spaces $P_k(I_{i,n})$ of real-valued functions **TODO: ES: why "real-valued"? i don't think there is any particular restriction on the value. complex numbers seem fine, and in fact, any vector space seems fine as well. can we just omit this? We should state it, otherwise the inner product would have to be redefined to have a complex conjugate. Like you said, we lose nothing by restricting to real-valued functions.** that, when restricted to $I_{i,n}$ become polynomials of degree less than k **TODO: ES: "less than" or "less than or equal to"? Less than. This way the space is k-dimensional and the sums and bounds look cleaner (instead of having $(k+1)$s everywhere)**, and are zero everywhere outside.

This leads to a space of basis functions on $[0,1]$, $\mathbf{V}_{n,k}$ defined as:

$$\mathbf{V}_{n,k} := \bigoplus_{1\le i\le 2^n} P_k(I_{i,n}). \tag{20}$$

Note that $\dim \mathbf{V}_{n,k} = k\,2^n$.

As before, $\mathbf{V}_{n-1,k} \subseteq \mathbf{V}_{n,k}$, and so we can again construct a hierarchical decomposition by choosing a complement to $\mathbf{V}_{n-1,k}$ in $\mathbf{V}_{n,k}$. In fact, we can obtain this decomposition canonically by using an inner product on the space of piecewise polynomial functions. For f, g in this space, define:

$$\langle f, g\rangle := \int_{[0,1]} f(x)g(x)\,dx. \tag{21}$$

Then letting $\mathbf{W}_{n,k}$ be the orthogonal complement to $\mathbf{V}_{n-1,k}$ in $\mathbf{V}_{n,k}$,

$$\mathbf{V}_{n,k} = \mathbf{V}_{n-1,k} \oplus \mathbf{W}_{n,k}, \qquad \mathbf{W}_{n,k} \perp \mathbf{V}_{n-1,k}. \tag{22}$$

From this we can construct a hierarchical orthonormal basis of functions for the entire space $\mathbf{V}_{n,k}$:

$$v_{\ell,i,j}(x) \qquad 0 \le \ell \le n,\ 1 \le i \le 2^l,\ 1 \le j \le k. \tag{23}$$

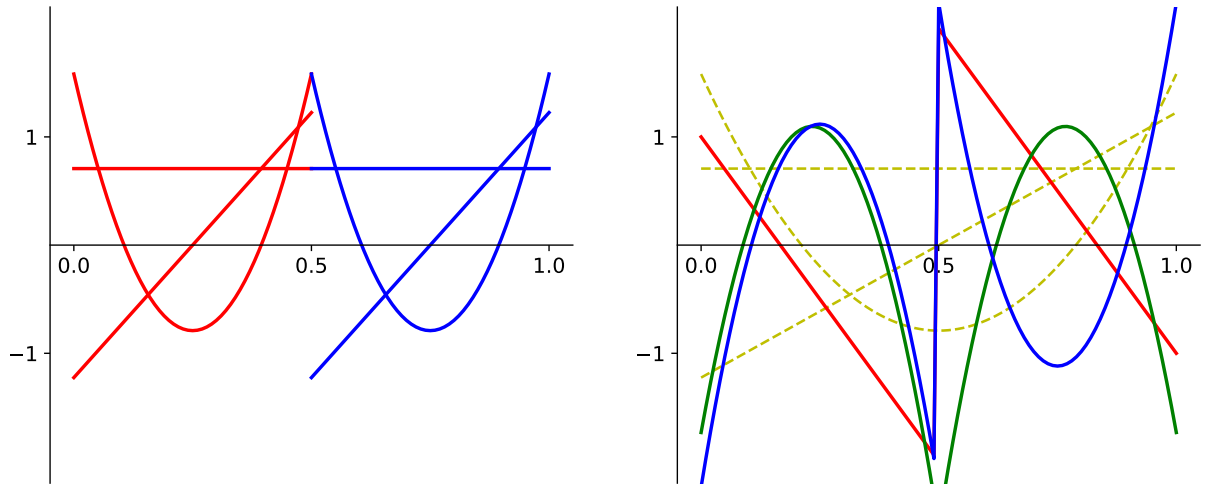

FIG. 6: Illustration of two different orthogonal bases for resolving the two half intervals of $[0, 1]$ (corresponding to resolution level one). On the left are piecewise Legendre polynomials for the original discontinuous Galerkin (DG) basis. On the right are the zeroth and first levels of the multi-resolution basis as $k = 3$. Level zero modes consist of Legendre polynomials, shown as dashed yellow lines. Higher levels are shifts/scalings to the appropriate subintervals of the three level-one basis modes drawn in red, green, and blue.

TODO: ES: here we use v (lower case) to denote a basis, whereas V (upper case) stands for a vector space. did we introduce a notation for a basis before? i think not. should we have? can we use here the same terminology and notation as before when we compared the (full) spaces and introduced the hierarchical basis? Here ℓ is the level and i is the element at level l as before. **TODO: ES: i suggest above to use "mode" before, not element. my reasoning is that elements have a special coupling that we don't use above, hence i wouldn't use the term "element" for the FD basis at all.** m here is called the *mode* and indexes the k orthogonal polynomials on the subinterval corresponding to (l, i). For a more explicit construction of this basis, see [11]. **TODO: ES: do we really want to introduce a new syntax with semicolon between the indices here? before, we just used commas to separate the level from the mode. if we want to use a semicolon, then we should do that everywhere, even when (for the FD basis) there is no element index. I removed it** Figure 7 illustrates resulting polynomials for a polynomial order $k = 3$. **TODO: ES: choose a consistent letter for the polynomial order that isn't used elsewhere. we can also use it in the FD section where there is a short paragraph speaking about higher-order finite differencing. i would use the letter p for that (lower case), which is standard notation. if so, we might need to use a different letter for the total number of basis functions P (upper case) to avoid confusion – maybe use Z instead? I'll have to think about this... I don't want to deviate too much from Wang's paper (where he uses k)**

In the d-dimensional case, we progress as before to apply the tensor product construction

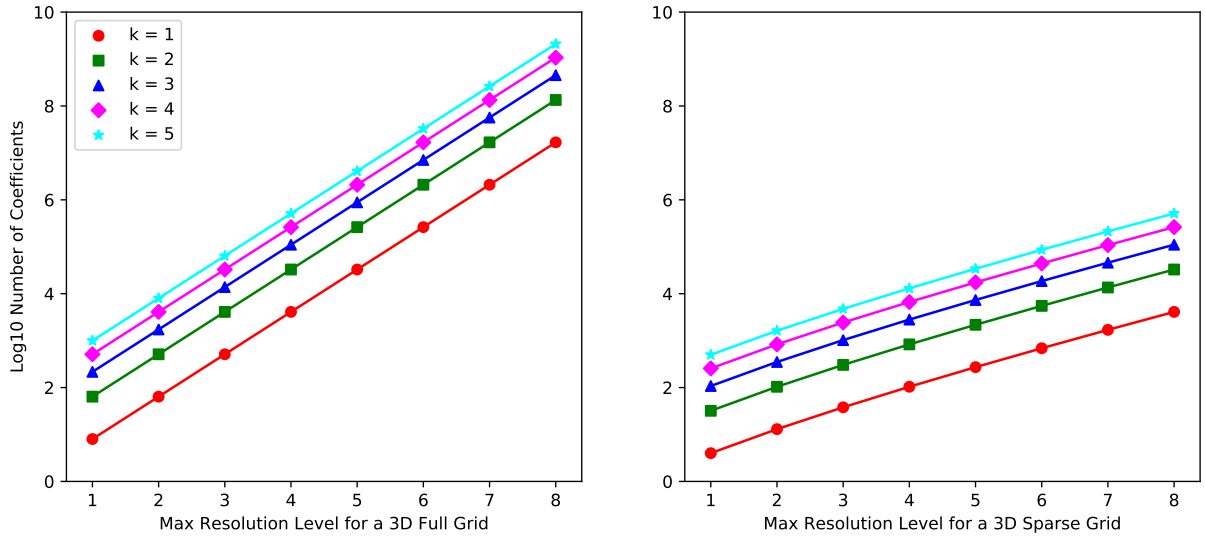

FIG. 7: The number of coefficients stored, on a log scale vs the maximum level of resolution, n, of a degree k Galerkin scheme in three dimensions. Shown for both the full and sparse spaces, i.e. $\dim \mathbf{V}_{n,k}$ and $\dim \hat{\mathbf{V}}_n^k$ for $d = 3$.

with the conventions of the previous Section II A:

$$v_{\mathbf{l},\mathbf{i},\mathbf{m}}(\mathbf{x}) = \prod_{d \le D} v_{\mathbf{l}_d,\mathbf{i}_d,\mathbf{m}_d}(\mathbf{x}_d), \tag{24}$$

$$\mathbf{W}_{\mathbf{l}}^k = \bigotimes_{d \le D} \mathbf{W}_{\mathbf{l}_d}^k. \tag{25}$$

The basis $v_{\mathbf{l},\mathbf{i},\mathbf{j}}$ spans the full discontinuous Galerkin grid space in d-dimensions. **TODO: ES: does it? isn't it more general in that it also allows for disconuities? that would be an important point. poor elaboration on my part: I meant the full-grid DG space. Fixed.** As before, $\mathbf{l}, \mathbf{i}$ are d-tuples of levels and elements called the multi-level and multi-element. $\mathbf{m}$ is a D-tuple of modes called a *multi-mode*. We can write this space in terms of its level decomposition as:

$$\mathbf{V}_{n,k}^D = \bigoplus_{|\mathbf{l}|_\infty \le n} \mathbf{W}_{\mathbf{l}}^k. \tag{26}$$

The corresponding sparse space is then defined as before [5, 6] by using the 1-norm to select only a subset of basis functions:

$$\hat{\mathbf{V}}_{n,k}^D = \bigoplus_{|\mathbf{l}|_1 \le n} \mathbf{W}_{\mathbf{l}}^k. \tag{27}$$

We will use this space in our tests and examples to efficiently construct solutions for hyperbolic PDEs in the next section.

B. Comparing Costs

Analogous to Equations (14) and (15) in section II B, we describe here the cost (number of basis functions) and errors associated with full and sparse DG spaces.

$$\dim \mathbf{V}_{n,k}^D = O(2^{nD}k^D) = O(N^D k^D) \tag{28}$$

$$\dim \hat{\mathbf{V}}_{n,k}^D = O(2^n k^D n^{D-1}) = O(Nk^D \log^{D-1} N). \tag{29}$$

This shows that a sparse grid reduces the cost significantly as the resolution N is increased, while our construction has no effect on the way the cost depends on the polynomial order k. **TODO: ES: is this something to investigate as well? would that make sense? this would be quite interesting, as it would directly translate to (pseudo-)spectral methods as well. if we can't easily say something, the we should at least point to "further investigation seems worthwhile" or similar. We looked at this for a bit, but there didn't seem to be any advantage to doing some sort of similar "sparse construction" in the space of polynomial degrees. It was also attempted in the Wang paper, I believe, but to poor results**

Further, from [12] we obtain the DGFE analogues of Equations (16) and (17) for the L^2-norm of the approximation error:

$$||u(\mathbf{x}) - \tilde{u}_{\text{full}}(\mathbf{x})||_2 = O(N^{-k}) \tag{30}$$

$$||u(\mathbf{x}) - \tilde{u}_{\text{sparse}}(\mathbf{x})||_2 = O(N^{-k} \log^{D-1} N). \tag{31}$$

which is equivalent to (16) and (17), except that we now achieve a higher convergence order k instead of just 2: The error found in sparse grids is only slightly (by a logarithmic factor) larger than that in full grids.

This yields the resulting ε-complexities:

$$\varepsilon_{L^2}^{\text{full DG}}(P) = O\left(P^{-k/D}\right) \tag{32}$$

$$\varepsilon_{L^2}^{\text{sparse DG}}(P) = O\left(P^{-k}\ k^{kD}\ \log^{(k+1)(D-1)} P\right) \tag{33}$$

TODO: ES: can we make a more concise statement, e.g. $O(kD)$, or "bounded by", or somesuch? I'll send you a note on what I've managed to get as a bound thus far. Update: I now have an exact bound for Equation (33) corroborated by Bungartz' paper This is consistent with an analogous estimate in [3]. Again, the sparse grid DG case provides a significant improvement over the full grid DG case.

C. Constructing the Derivative Operator

To solve hyperbolic PDEs, we must define a derivative operator on the multi-dimensional sparse basis. To do this, we proceed in steps:

1. Define the derivative on the one-dimensional Galerkin space $\mathbf{V}_{n,k}$ in terms of the standard non-hierarchical basis.

2. Express this operator in the one-dimensional hierarchical basis.

3. Using the tensor-product construction, define the derivative matrix elements in the full multi-dimensional basis $\mathbf{V}^D_{n,k}$.

4. Restrict this operator to the sparse subspace, i.e. only calculate matrix elements between basis functions in $\hat{\mathbf{V}}^D_{n,k}$, discarding those matrix elements that lead out of this subspace.

To define the derivative in the standard Galerkin basis in one dimension, we follow the methods outlined in a prior paper [10]. For a given resolution n **TODO: ES: what is "resolution"? is this the number of levels L? Yes.** and degree k, let $\{h_{i,m} : 1 \le i \le 2^n,\ 1 \le m \le k\}$ be a Galerkin basis for $\mathbf{V}_{n,k}$ so that for a given i, each $h_{i,m}$ is supported only on the interval $I_{i,n}$ from before, and together constitute a basis of k orthonormal polynomials of degree less than k on that interval. **TODO: ES: why are we constructing a new basis here? shouldn't we introduce it earlier, if it is needed? how does it differ from the basis v we introduced before? if this is simply about finding the basis within each element (i.e. finding the modes m), then we should have introduced them before. we even have a figure showing them, constructed via orthogonality to subspaces etc.! I see your point. The reason I'm working with legendre polynomials is simply because they're localized (unlike the multiresolution $v_{l;i,m}$'s, and so its easy to write out the derivative operator in this basis. It would be totally acceptable, however, to work in just the multiresolution basis and define the derivative by inner product)** In our implementation, the $h_{i,j}$ are chosen as shifts and scales of the restricted Legendre polynomials: **TODO: ES: why do we now need Legendre polynomials? They're the best orthonormal generalization of the nodal basis from before.**

$$h_{i,m}(x) = \begin{cases} (\sqrt{2})^{n+1} P_m\left(2^n \cdot (x - i \cdot 2^{-n})\right) & \text{if } x \in I_{i,n} \\ 0 & \text{otherwise.} \end{cases} \tag{34}$$

The derivative operator is then defined in the weak sense:

$$\int_{[0,1]} h_{i,m} \frac{df}{dx}\, dx = \int_{I_{i,n}} h_{i,m} \partial_x f\, dx = \Big[h_{i,m} f\Big]_{(i-1)2^{-n}}^{i2^{-n}} - \int_{I_{i,n}} f(x) \frac{dh_{i,m}}{dx}\, dx \tag{35}$$

Because the $h_{i,m}$ are discontinuous at the two boundary points, where they jump from zero to nonzero values, we define $h_{i,m}((i-1)\cdot 2^{-n}), h_{i,m}(i\cdot 2^{-n})$ to be the average of the two, namely half the endpoint value. **TODO: ES: this requires a more detailed description. i think we either need to argue carefully via distributions, or side-step this issue by saying things like "we choose this function g as representation of th derivative of f" and "a more detailed construction following [10] would show that". I can find a source, one moment** The boundary term then becomes:

$$\frac{1}{2} \lim_{\epsilon \to 0^+} \left[f\left(i2^{-n} - \epsilon\right) h_{i,m}\left(i2^{-n} - \epsilon\right) - f\left((i-1)2^{-n} + \epsilon\right) h_{i,m}\left((i-1)2^{-n} + \epsilon\right) \right] \tag{36}$$

while in the second integral, $h_{i,m}$ is treated exactly as a polynomial function. Its derivative is then exactly computable. By knowing the basis representation of f in terms of $h_{i,m}$, we can obtain an explicit matrix form of the derivative operator by writing

$$\mathcal{D}_{i,m;\, i',m'} := \int_{[0,1]} h_{i,m} \frac{dh_{i',m'}}{dx}\, dx. \tag{37}$$

This matrix will be a sum of two matrices, as above. One corresponding to the discontinuity terms at the boundary, and the second corresponding to the regular derivative operator on the smooth polynomial functions in the interior. **TODO: ES: i think we need to re-write the description above. i will do this after we changed the notation. How's this? j has changed to m for "mode"**

By orthogonally transforming $h_{i,j}$ into the hierarchical Galerkin basis $v_{l;i,j}$ through a transformation Q, we can conjugate **TODO: ES: "conjugate"? is this an established term? otherwise i'd call it "transform". I call it conjugate because QAQ^{-1} is referred to as the "conjugation action of Q on A". I can use transform, too.** this derivative operator into its hierarchical form $D^{\text{hier}} = QDQ^T$. **TODO: ES: i guess this implies that Q is orthonormal? is it? why not write Q^{-1} instead of Q^T? Yes, because both bases (position-based Legendre and multiresolution) are orthonormal.** We generalize this operator to higher dimensions through the standard tensor product construction. The partial derivative operator ∂_a, $1 \le a \le d$, is represented by the derivative operator $\mathcal{D}_a^{\text{full}}$ acting on the basis $v_{\mathbf{l};\mathbf{ij}}$ as

$$\mathcal{D}_a^{\text{full}}(v_{\mathbf{l};\mathbf{ij}}) = \left[\mathcal{D}^{\text{hier}} v_{\mathbf{l}_a;\mathbf{i}_a\mathbf{j}_a}\right] \prod_{m\neq a} v_{\mathbf{l}_m;\mathbf{i}_m\mathbf{j}_m}. \tag{38}$$

The last step is to restrict this full derivative operator to the sparse space. We obtain $\mathcal{D}_a^{\text{sparse}}$ from $\mathcal{D}_a^{\text{full}}$ by restricting the latter to the sparse basis space. Note, however, that $\mathcal{D}_a^{\text{full}}$ does not preserve the sparse space, as it has nonzero matrix entries mapping sparse to non-sparse basis elements. Note that this is also the case for the 1D derivative operator $\mathcal{D}^{\text{hier}}$, which doesn't stabilize $\mathbf{V}_{n,k} \subset \mathbf{V}_{n+1,k}$. Despite this, when restricting the derivative matrix to $\mathbf{V}_{n,k}$, we ignore those components of the operator that map outside of $\mathbf{V}_{n,k}$. We shall do the same here, and define

$$\mathcal{D}_a^{\text{sparse}} = \text{Proj}_{\hat{\mathbf{V}}_{n,k}^d} \mathcal{D}_a^{\text{full}}. \tag{39}$$

This will be the working definition for our derivative operator in the sparse basis.

For PDEs with periodic boundary conditions, the 1D derivative operator, from which the full and sparse derivatives are constructed, will have the boundary terms in the integration by parts (35) overlap at the opposite endpoints 0 and 1. This turns the interval $[0,1]$ into the torus T^1.

D. Evolving a Scalar Wave in the DGFE Basis

Using the sparse derivative operator that we have constructed, we turn to solving the wave scalar equation (in a flat spacetime)) as example and test case. This equation can be reduced to first order in time as

$$\begin{aligned} \dot{\varphi}(\mathbf{x},t) &= \psi(\mathbf{x},t) \\ \dot{\psi}(\mathbf{x},t) &= \nabla^2\varphi(\mathbf{x},t) \end{aligned} \quad \Rightarrow \quad \frac{d}{dt}\begin{pmatrix}\varphi\\ \psi\end{pmatrix} = \begin{pmatrix}0 & \mathbf{1}\\ \nabla^2 & 0\end{pmatrix}\begin{pmatrix}\varphi\\ \psi\end{pmatrix}. \tag{40}$$

We then use the DGFE basis to represent this as

$$\varphi(\mathbf{x},t) = \sum_{\mathbf{l};\mathbf{i},\mathbf{j}} \varphi_{\mathbf{l};\mathbf{i},\mathbf{j}}(t)\, v_{\mathbf{l};\mathbf{i},\mathbf{j}}(\mathbf{x}), \qquad \psi(\mathbf{x},t) = \sum_{\mathbf{l};\mathbf{i},\mathbf{j}} \psi_{\mathbf{l};\mathbf{i},\mathbf{j}}(t)\, v_{\mathbf{l};\mathbf{i},\mathbf{j}}(\mathbf{x}), \tag{41}$$

$$\nabla^2 \to \sum_a \mathcal{D}_a^{\text{sparse}} \cdot \mathcal{D}_a^{\text{sparse}}. \tag{42}$$

As written, this representation is exact – it only corresponds to a particular basis choice. We then introduce a cut-off for the number of levels L, which introdcues an approximation error.

Here, $\varphi_{\mathbf{l};\mathbf{i},\mathbf{j}}, \psi_{\mathbf{l};\mathbf{i},\mathbf{j}}$ are the coefficients corresponding to the interpolations of φ, ψ. The problem is now framed as a set of coupled first-order ordinary differential equations (ODEs). Such ODEs can be solved in a straightforward manner via a wide range of ODE integrators, e.g. of the Runge-Kutta family.

Since the derivative and Laplacian operators are sparse matrices, and since the maximum allowable timestep scales with the linear problem size N due to the Courant-Friedrichs-Lewy criterion, the overall cost to calculate a solution scales as $O(N^2 \log^{D-1} N)$, which is significantly more efficient than the $O(N^{D+1})$ scaling for a full grid. Here D is the number of spatial dimensions. In principle it would be possible to employ a sparce space-time discretisation as well, reducing the cost further to $O(N \log^D N)$. However, this would require a more complex time evolution algorithm, and we thus do not pursue this appraoch here.

Initial conditions for this time evolution are given by the coefficients of the initial position and velocity of the wave φ_0, ψ_0. This requires projecting given functions into the sparse space. (Later, time evolution proceeds entirely in coefficient space.) Projecting onto the basis functions requires (in practice) a numerical integration for each coefficient. If one is satisfied with the accuracy and convergence properties of sparse grids, then one can replace this numerical integration with sampling the given function at the collocation points of the DG basis functions, as is often done. However, we do not want to make such assumptions in this paper, and thus use a (more accurate) numerial integration to properly project the initial conditions onto the basis functions.

It is advantageous (the calculation is simpler) if the initial conditions can be cast as sums of tensor products of 1D functions. By projecting these individual components of the tensor in 1D, and the applying the tensor construction to those 1D coefficients, we can recover the coefficients of the full d-D function. This yields a considerable speedup as all integrations only need to be performed in 1D.

Such a construction can in particular be applied to all sine waves $\sin(\mathbf{k} \cdot (\mathbf{x} - \mathbf{x}_0))$ by expressing them as sums of products of sines and cosines in one variable. In the general case, a sparse integrator (e.g. based on sparse grids!) could be used to avoid concerns with the cost of high-dimensional integration.

We present results in section V below, after briefly introducing the Julia package we developed to perform these calculations.

IV. OVERVIEW OF THE OPEN-SOURCE PACKAGE

The Julia code for performing interpolation and wave evolution using the sparse Galerkin bases outlined in this paper is publicly available as open source under the "MIT" licence at

https://github.com/ABAtanasov/GalerkinSparseGrids.jl

To obtain the basis coefficients resulting from projecting a function `f` in `D`-dimensions at degree `k` and level `n`, we use the command `coeffs_DG` as follows:

```
using GalerkinSparseGrids
D = 2; k = 3; n = 5;
f = x -> sin(2*pi*x[1])*sin(2*pi*x[2])
```

```
full_coeffs   = coeffs_DG(D, k, n, f; scheme="full"  )
sparse_coeffs = coeffs_DG(D, k, n, f; scheme="sparse")
```

Obviously, calling `full_coeffs` is much more expensive than `sparse_coeffs`, and should only be used to compare cost and accuracies.

This create a dictionary-like data structure for easy access to coefficients at the various levels and elements. Conversely, given such a dictionary of coefficients, we can evaluate the function at a `D`-dimensional point `xs` using `reconstruct_DG`.

```
full_val   = reconstruct_DG(full_coeffs  , xs)
sparse_val = reconstruct_DG(sparse_coeffs, xs)
```

For algorithms such as time evolution and differentiation, these coefficient tables are converted into vectors (one-dimensional arrays), so that the derivative operators act as multiplication by a sparse matrix, and all operations can be performed efficiently.

To solve the wave equation, we provide the following function:

```
wave_evolve(D::Int, k::Int, n::Int,
                f0::Function, v0::Function,
                time0::Real, time1::Real;
                order="45", scheme="sparse")
```

Using the representation of the intial data and Laplacian operator given in Equations (41) and (42), this represents the D-dimensional wave equation with initial conditions `f0` for φ and `v0` for ψ by a set of linearly coupled first order ODEs. This can then be solved by any integrator in the various Julia packages for ordinary differential equations. In the present package, we chose to use ode45 and ode78 from the 'ODE.jl' package [13].

For example, to solve a standing wave in 2D from time `t_0` to time `t_1` with polynomials of degree less than `k` on each element, using a sparse depth `n`, and with the `ode78` ODE integrator:

```
f0 = x->sin(2*pi*x[1])*sin(2*pi*x[2])
v0 = 0
sparse_soln = wave_evolve(2, k, n, f0, v0, t_0, t_1; order="78")
```

TODO: ES: can you add more text here? it should be easy to extend this section from one to (say) three pages, giving more details or more hand-holding, or describing more about how the data are organized. this will make people feel more comfortable with the package and make it more likely that they will adopt it. for example, you could also walk people through solving a 1D problem (including plotting results), or walk them through the code used to create one of the data points in one of the error figures, etc.

More documentation is available in the Github repository and as part of the source code.

In the next section we use these methods to demonstrate how the sparse Galerkin basis allows for accurate and efficient time evolution in higher dimensions.

V. RESULTS

The Galerkin bases and derivative operators constructed above allow representing arbitrary functions, calculating their derivatives, solving PDEs, and evaluating the solution at arbitrary points, as is the case with any other discretization method.

We focus here on the scalar wave equation as simple example application. Using the constructions developed the prior sections, we use the Galerkin sparse grid package to calculate efficient and accurate representations in 3D and 5D, as well evolving the wave equation in time in 3+1 and 5+1 dimensions. At this accuracy, this would have been prohibitively expensive using full grid methods.

We note that, in general, spectral methods can offer superior accuracy, but they also suffer from the curse of dimensionality, and are limited to representing smooth functions; representing non-smooth functions requires high orders. DGFE discretizations combine the advantages of both local (e.g. finite differencing) and global (spectral methods), hence our interest in DG sparse grids. If only a single spectral level is used, then DG sparse grids are equivalent to a spectral method. DG sparse grids are thus a true superset of spectral discretizations.

In the following discussion, all errors have been calculated via a Monte Carlo integration to evaluate integrals of the form

$$\int_{[0,1]^d} |u^*(\mathbf{x}) - \tilde{u}(\mathbf{x})|^2 \, d^d\mathbf{x} \tag{43}$$

for u^* the analytically correct solution and $\tilde{u}$ the DGFE solution. The L^2 error is output as the square root of the above quantity. We use approximately 1000 points, enough to guarantee convergence to satisfactory precision for error measurement in our case.

A. Approximation Error for Representing Given Functions

We begin by evaluating the approximation error made when representing a given function with a sparse basis at a particular cut-off level. We choose as function a sine wave of the form

$$f(\mathbf{x}) \;=\; A\cos(\mathbf{k}\cdot\mathbf{x} + \phi) \tag{44}$$

with $A = 1.3, \phi = 0.4$, with $\mathbf{k} = [1, 2, -1]$ in 3D and $\mathbf{k} = 2\pi \cdot [1, 0, -1, 2, 1]$ in 5D. We also employ periodic boundary conditions. These first tests do not yet involve derivatives or integration.

Figure 8 compares the accuracy achieved with a particular number of coefficients for full and sparse grids. As expected, sparse grids are more efficient (in the sense of the accuracy/cost ration), and this efficiency increases for higher resolutions. As also expected, this effect is even more pronounced in the $D = 5$ case.

TODO: ES: these should be labelled "error vs. level" and "error vs. number of coefficients" instead TODO: ES: can you show full grid results for comparison? a single k value should suffice ($k = 5$?); i expect the error to be really bad.

Even in two dimensions, the sparse Galerkin basis already provides a considerable advantage in efficiency. For the sake of visual intuition, we show the approximation error of representing a sine wave in 2D by Galerkin sparse grids of in Figure 9, for polynomial order $k = 3$ and sparse level $n = 5$.

B. Approximation Error in Time Evolution

Beyond just representing given functions, Figures 10 and 11 demonstrate the accuracy of evolving a wave evolution using DGFE sparse grids. After projecting the initial position

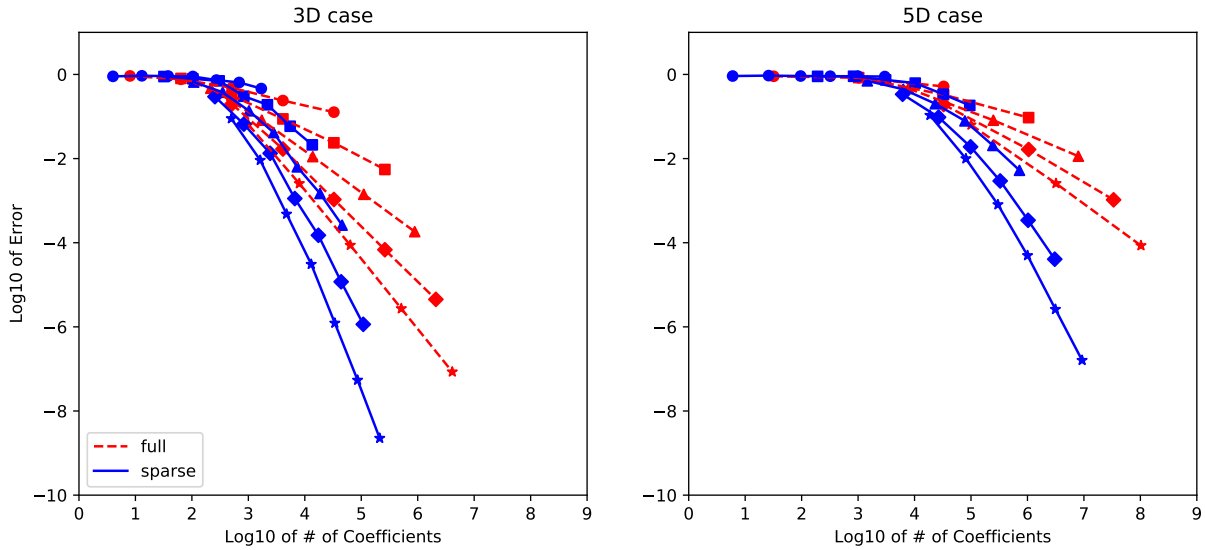

FIG. 8: Comparing approximation errors as a function of the number of coefficients for $D = 3$ (left) and $D = 5$ (right) for values of the polynomial order k from 1 to 5. Even at moderate scales (10^5 coefficients), sparse grids for $k = 5$ exhibit errors as much as $1,000$ times smaller than their full counterparts. Particularly noteworthy here is the fact that the accuracy gain from higher polynomial order k in the sparse scheme is far more significant than in the full scheme. This shows significant promise for higher-order sparse Galerkin methods. See main text for more details.

and velocity of a travelling wave in 3 and 5 dimensions onto the sparse basis, we evolved the system in time using the derivative matrices $\mathcal{D}_a^{\text{sparse}}$ and wave evolution scheme developed in Sections III C and III D, respectively.

Initial conditions for the evolution are

$$\varphi(\mathbf{x}, 0) = \cos(\mathbf{k} \cdot \mathbf{x} + \phi), \;\; \psi(\mathbf{x}, 0) = -\omega \sin(\mathbf{k} \cdot x + \phi)$$

where here $\mathbf{k}$ is a wavenumber vector compatible with the periodic boundary conditions.

In summary:

AA: Let me know anything else that you'd like me to add. I could write up a more general Klein-Gordon solver or even try to do an elementary strong gravity problem, otherwise I'll save that for future updates of the package.

TODO: ES: Can you take the 5D solution for the wave equation evolution and extract a 2D plot similar to figure 9? you could e.g. set $x_2 = x_3 = x_4 = 0$ and choose both $t = 0$ and $t = 1.32$ crossing times.

VI. CONCLUSION

TODO: ES: i will write this, and also add acknowledgements

[1] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

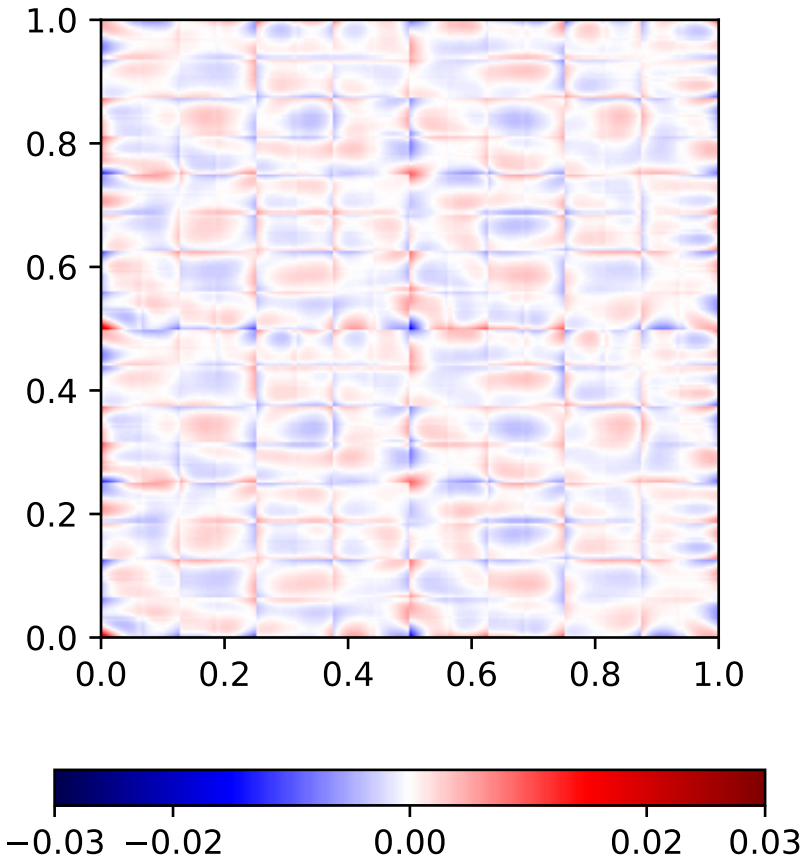

FIG. 9: Approximation error (difference to exact value) of a sine wave $\varphi(\mathbf{x}) = A\cos(\mathbf{k}\cdot\mathbf{x}+\phi)$ on $[0,1]\times[0,1]$ with $A = 1.3, \phi = 0.4$, and $\mathbf{k} = 2\pi\,(1,2)$ in 2D for $k = 3,\ n = 5$. The small discontinuities typically introduced at element boundaries and corners by DGFE methods are clearly visibible. Contrary to the naive assumption, the error is *not* the largest in the interior (away from the boundaries), where the sparse grid structure has removed collocation points.

[2] Sergey A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 4:240–243, 1963.

[3] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta numerica*, 13:147–269, 2004.

[4] Thomas Gerstner and Michael Griebel. *Sparse Grids*. John Wiley & Sons, Ltd, 2010.

[5] Wei Guo and Yingda Cheng. A sparse grid discontinuous galerkin method for high-dimensional transport equations and its application to kinetic simulations. *SIAM Journal on Scientific Computing*, 38(6):A3381–A3409, 2016.

[6] Zixuan Wang, Qi Tang, Wei Guo, and Yingda Cheng. Sparse grid discontinuous galerkin methods for high-dimensional elliptic equations. *J. Comput. Phys.*, 314(C):244–263, June 2016.

[7] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

[8] Christoph Zenger. *Sparse Grids*, volume 31. Friedrich Vieweg & Sohn Verlagsgesellschaft, 1991.

[9] Joseph Traub and Henryk Wozniakowski. A general theory of optimal algorithms / j. f. traub, h. wozniakowski. 01 1980.

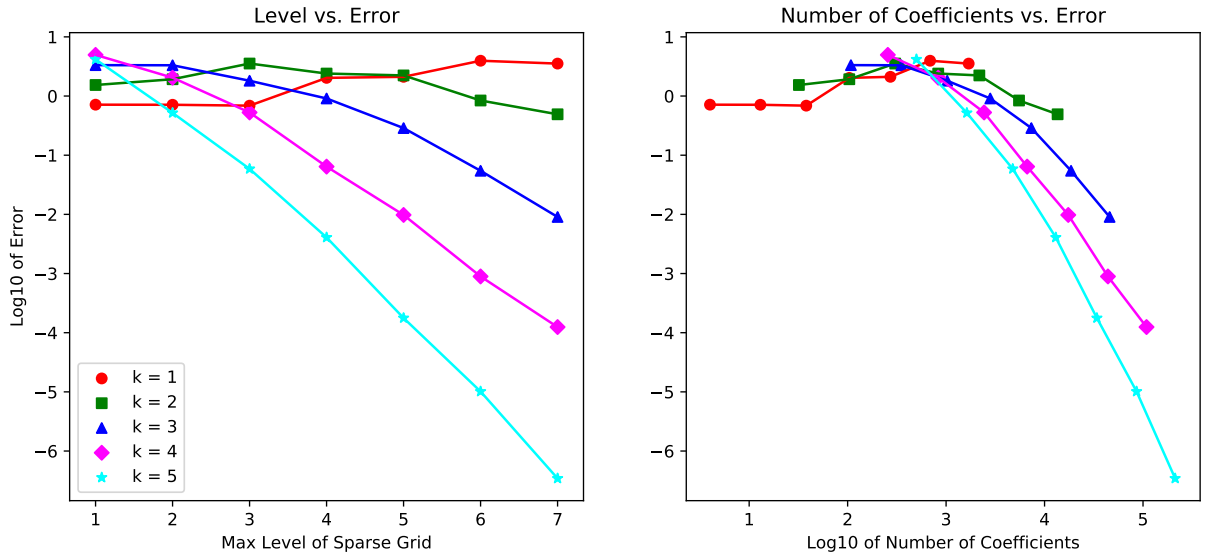

FIG. 10: Error of travelling wave evolution for 1.32 crossing times in 3D, using $\mathbf{k} = (1, 2, -1)$ from $t = 0$ to $t = 0.54$ at various levels and degrees of interpolation. Periodic boundary conditions.

TODO: ES: can you show full grid results for comparison? Yes, this will be what I'll use the yale HPC cluster for. Perhaps we should forget the error vs. level plots altogether and make side-by-side 3D sparse vs. full overlay (left), 5D sparse vs. full overlay (right) as in Figure 8

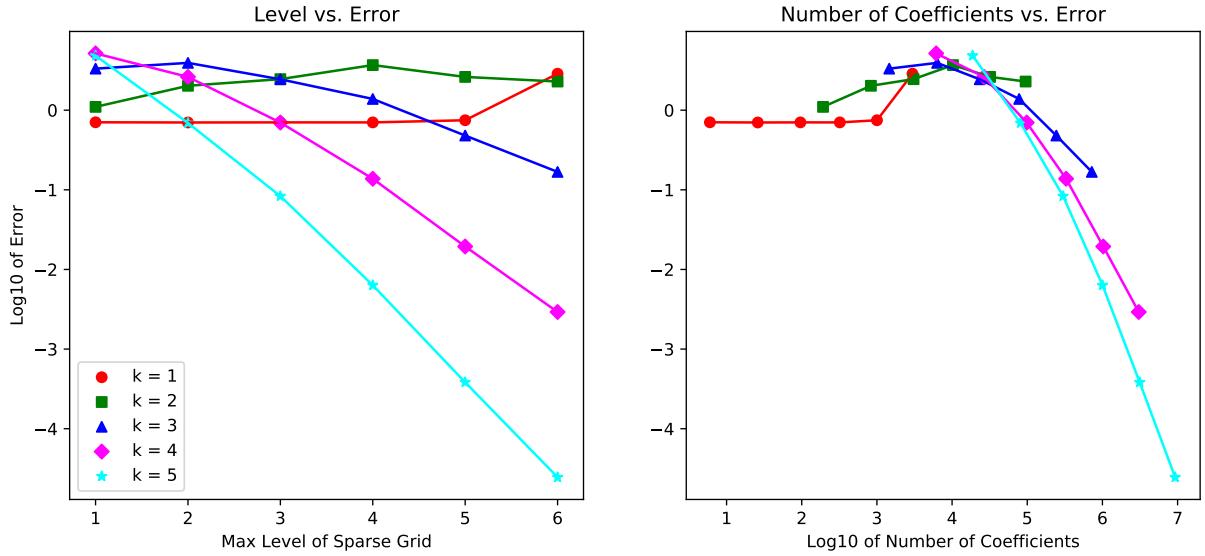

FIG. 11: Error of travelling wave evolution for 1.43 crossing times in 5D, using $\mathbf{k} = (1, 0, -1, 2, 1)$ from $t = 0$ to $t = 0.54$ at various levels and degrees of interpolation. Periodic boundary conditions.

[10] Jonah M Miller and Erik Schnetter. An operator-based local discontinuous galerkin method compatible with the bssn formulation of the einstein equations. *Classical and Quantum Gravity*, 34(1):015003, 2017.

[11] Bradley K. Alpert. A class of bases in l2 for the sparse representations of integral operators. *SIAM J. Math. Anal.*, 24(1):246–262, January 1993.

[12] Schwab, Christoph, Sli, Endre, and Todor, Radu Alexandru. Sparse finite element approximation of high-dimensional transport-dominated diffusion problems. *ESAIM: M2AN*, 42(5):777–819, 2008.

[13] ODE.jl. 2017-09-23. 0.7.0. https://github.com/JuliaDiffEq/ODE.jl.