Summary of Summer Research at Perimeter, 2016

Alex Atanasov

This project involves the use of the Sparse Grid construction, originally introduced by Smolyak, to find ways of getting around the curse of dimensionality for high dimensional PDEs. In particular, we make use of the discontinuous Galerkin method in 1-D and generalize it to be compatible with sparse grid constructions. We then apply this construction to solving hyperbolic PDEs in 2-D.

WLOG we will work on the interval $\Omega = [0, 1]$ or the appropriate "unit hypercube" $\Omega = [0, 1]^d$.

1 The Sparse Grid Construction in the Hat Basis
(Mid-May thru Mid-June)

During the first month of my research, I worked in the **hat basis**, where all my basis functions were shifted and rescaled copies of the simple function

$$\phi(x) = \begin{cases} 0, & \text{if } |x| > 1 \\ 1 - |x|, & \text{otherwise} \end{cases}. \tag{1}$$

This hat function, ϕ, gives us a canonical **"position"** or **Lagrange basis** of resolution 2^l by considering all translates:

$$\phi_i^l(x) = \phi(2^l(x - i2^{-l})) \tag{2}$$

for i running from 0 to 2^n.

We can approximate a function f within this finite dimensional basis as

$$f(x) \approx \sum_i a_i \phi_i^l(x) \tag{3}$$

$$a_i = f(i2^{-l}) - \frac{1}{2} f((i-1)2^{-l}) - \frac{1}{2} f((i+1)2^{-l}). \tag{4}$$

The span of this position basis is denoted by V_l

This position basis approximation generalizes easily to higher dimensions by considering **tensor product constructions** of the ϕ_i^l. This spans a space denoted by $V_{\vec{l}}$ of resolution l_j along dimension j. An implementation can be found in the Mathematica notebooks.

For the purpose of this project, we must work in a **multiresolution basis**, where instead of working at a specific l, we interpolate f at multiple different levels k from $k = 0$ to some final l. We then have a basis $\phi_{k,i}(x)$ defined by

$$\phi_{k,i}(x) = \phi\left(2^k\left(x - 2^{-k}(2i+1)\right)\right) \tag{5}$$

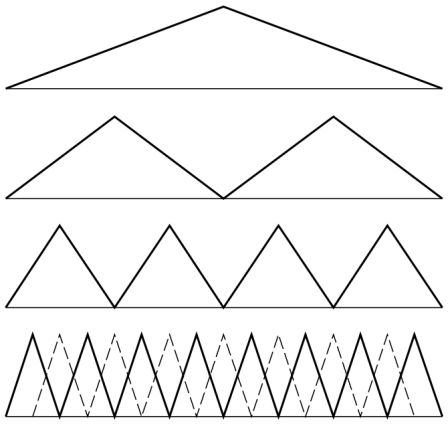
This basis spans the same space as the position basis, namely V_l. We can in fact see that this multiresolution construction takes advantage of the fact that:

$$V_0 \subset V_1 \subset \cdots \subset V_{l-1} \subset V_l \tag{6}$$

Each new set of basis functions is in the orthogonal complement in V_k to V_{k-1}. We will denote this by W_k so that:

$$V_l = \bigoplus_{j=0}^{l} W_j \tag{7}$$

A pictorial representation of this multiresolution basis is below:



With this multiresolution basis, again implemented in the mathematica notebooks, we can also employ a **tensor product construction** for higher dimensions. It is this tensor product space that we will "cut out" the subspaces that have very high dimension but are not important to having a good resolution.

For a tensor space in d-dimensions, we have elements:

$$\phi_{\vec{k},\vec{i}}(\vec{x}) = \prod_j \phi_{k_j,i_j}(x_j) \tag{8}$$

If we wish to span $V_{\vec{l}}$ then we must have each k_i run from 0 to the appropriate l_i.

Defining $W_{\vec{k}}$ as $\text{span}_{\vec{i}}\{\phi_{\vec{k},\vec{i}}\}$, we again get

$$V_{\vec{l}} = \sum_{\vec{k} \leq \vec{l}} W_{\vec{k}} \tag{9}$$

where $\vec{a} \leq \vec{b}$ means $\forall i,\ a_i \leq b_i$. The dimension of this space is $O(2^{|\vec{l}|_1})$ or if $|l|_1 = (n, n, \ldots, n)$ then it is $O(2^{nd})$

We can now form the **sparse grid space**, defined as

$$Q_n = \bigoplus_{|\vec{k}|_1 \leq n} W_{\vec{k}}. \tag{10}$$

The dimension of this space is $O(2^n n^{d-1})$. It is a significant reduction in dimensionality from $O(2^{nd})$ for a full grid of $\vec{l} = (n, n, \ldots, n)$, with in fact only a n^{d-1} factor increase in the error.

This construction is done in 2-D and in general d-D in the Mathematica notebooks. Explicit plots of error comparisons are done there as well.

2 The Multiresolution Discontinuous Galerkin (DG) Basis

(Mid-June thru Early-July)

For implementing the discontinuous Galerkin method, the canonical way forward is to divide the interval $\Omega = [0, 1]$ into sub-regions $\Omega_i = [hi, h(i+1)]$ just like with the position basis (where $h = 2^{-l}$). On each subinterval, now, we use **polynomial basis functions** up to a certain degree p instead of hat functions. These polynomials are only supported on Ω_i, and vanish outside of it (thus causing the discontinuities that give this method its name). We call this an h-p **discretization scheme**. A common choice is Legendre or Tchebychev polynomials for a basis, as this way the coefficients can be extracted immediately from a simple integration.

For higher dimensions, it is common to apply a tensor product construction to these basis functions. Such a basis will face the curse of dimensionality, since there is no way to apply sparse grid techniques to such a position basis. For this reason, we must go into a **multiresolution DG basis**, defined analogously to before.

We begin at resolution zero, with the Legendre polynomial basis up to degree p on the whole interval Ω. At resolution one, we need to find basis functions defined piecewise on the two subintervals that are orthogonal to the $k = p+1$ Legendre polynomials above. Because of these orthogonality considerations, the basis functions at the next level will satisfy certain symmetry properties about $x = 1/2$.

We will consider shifts and scalings of the following functions:

$$\psi_j(x) = \begin{cases} 0, & \text{if } |x| > 1 \\ x^j, & \text{if } x > 0 \\ (-1)^j x^j, & \text{if } x < 0 \end{cases} . \tag{11}$$

from $i = 0$ to p. To find the level k contribution to the multiresolution DG basis we work with the space spanned by $\psi_j(2^k(x - i2^{-k}))$ for $i = 1$ to $2^k - 1$ At each interval Ω_i at resolution k, we consider the functions $\psi_j(2^k(x - i2^{-k}))$ for that specific i and consider all possible j. We pick out the subspace that has the first p moments vanish. This means that we are orthogonal to all polynomials up to degree p, which is precisely what the lower-resolution functions restruct to on Ω_i. This orthogonalization process has been explicitly implemented in the Julia package GalerkinSparseGrids.jl.

Now with this multiresolution basis, we can perform the sparse construction as before. The only caveat is that since we have built orthogonality up with respect to the standard $L^2(\Omega)$ norm, when extracting coefficients through an integration, we need a quadrature process that does not suffer the curse of dimensionality.

For this reason, functions such as hcubature will not work. We would instead want a Smolyak-type quadrature integrator (which have already been implemented). This is not part of the scope of the project, so for now we are ignoring this issue.

3 Defining a Multiresolution Derivative Operator Evolving a 1-D Wave equation

(Early-July thru Late-July)

The derivative operator for the DG method is formulated in the weak sense. For a basis function $v_i(x)$, we want to find the matrix element of the derivative operator given by

$$D_{ij} = \int_\Omega v_i(x) D(v_j(x)) dx \tag{12}$$

but since v_j is defined specifically on a sub-interval Ω_s this becomes:

$$\int_{\Omega_s} v_i D(v_j) dx = [v_i v_j]_{\partial\Omega_s} - \int_{\Omega_s} D(v_i) v_j \tag{13}$$

where the last integral is done properly on the interior of Ω_s. For the boundary term, because of discontinuity the common choice is to define v_i on the boundary as

$$\{v_i\}(p) = \frac{1}{2}(v_i(p-\varepsilon) + v_i(p+\varepsilon)) \tag{14}$$

$$\Rightarrow [v_i v_j]_{\partial\Omega_s} = \{v_i\}(+)v_j(+) - \{v_i\}(-)v_j(-) \tag{15}$$

There are many references that advise for different ways of treating the derivative. This is just one of the ways. It may turn out that this is not the optimal way to work with the derivative for this DG sparse grid construction.

With this way of defining the derivative on our basis, we can define it for the functions we interpolate:

$$Df \approx \sum_j a_j (Dv_j) = \sum_j a_j D_{ij} v_i \tag{16}$$

This naturally gives us a laplacian $D(Df)$. We can solve the wave equation $\ddot{f} = \nabla^2 f$ by writing it as two coupled first order equations:

$$\dot{f} = \rho \tag{17}$$

$$\dot{\rho} = D(Df) \tag{18}$$

Because of our discretization, this is now an linear algebraic ordinary differential equation. It can be solved using an appropriate ODE solver (c.f. ODE.jl).

Efficient sparse-matrix representations of the derivative have been implemented in the Julia package GalerkinSparseGrids.jl.

By simply calculating the matrix corresponding to a change of basis from the regular DG basis in 1-D to the multiresolution 1-D DG basis, we can find the explicit matrix form for the derivative in the multiresolution basis as well (without having to worry unnecessarily about boundary terms and comparisons, which is more complicated for the multiresolution basis). We directly have:

$$D^{\text{hier}} = Q^{-1}_{\text{hier}\leftarrow\text{pos}} D^{\text{pos}} Q_{\text{pos}\leftarrow\text{hier}} = Q^{T}_{\text{hier}\leftarrow\text{pos}} D^{\text{pos}} Q_{\text{pos}\leftarrow\text{hier}} \tag{19}$$

since both bases are orthogonal, Q is an orthogonal matrix. This matrix is the hier2pos method in the Position_Basis_DG.jl script.

4 Defining a Multiresolution Derivative Operator Evolving a 2-D/n-D Wave equation using Sparse Grids

(Late-July thru Present)

With the derivative defined in the 1-D multiresolution basis, it should be easy to extent to to higher dimensions on the multiresolution tensor product space. That is, for a multidimensional basis function $V_{\vec{l}\vec{p}}^{\vec{f}}$ where $\vec{l}$ is the multi-level, $\vec{p}$ is the multi-place at that level, and $\vec{f}$ is the multi-function number, we get for the derivative D_1 with respect to x_1 that:

$$(D_1^{hier})_{ij} = \left\langle V_i D_1^{\text{hier}}(V_j) \right\rangle = \left\langle V_{\vec{l}_1,\vec{p}_1}^{\vec{f}_1} D_1^{\text{hier}} \left(V_{\vec{l}_2,\vec{p}_2}^{\vec{f}_2} \right) \right\rangle \tag{20}$$

$$= \delta_{\vec{l}_1[2:],\vec{l}_2[2:]} \delta_{\vec{p}_1[2:],\vec{p}_2[2:]} \delta^{\vec{f}_1[2:]\vec{f}_2[2:]} \left\langle v_{\vec{l}_1[1],\vec{p}_1[1]}^{\vec{f}_1[1]} D_{\text{1D}}^{\text{heir}} v_{\vec{l}_2[1],\vec{p}_2[1]}^{\vec{f}_2[1]} \right\rangle \tag{21}$$

$$= \delta_{\vec{l}_1[2:],\vec{l}_2[2:]} \delta_{\vec{p}_1[2:],\vec{p}_2[2:]} \delta^{\vec{f}_1[2:]\vec{f}_2[2:]} (D_{\text{1D}}^{\text{heir}})_{lpf,lpf'} \tag{22}$$

This is implemented in the Multidim_Derivative.jl script and is the full_D_matrix method. By including an if-statement to give us the appropriate sparse cutoff, we get the sparse_D_matrix method. As before, we can define a Laplacian by summing the squares of the D_i^{sparse} matrices over each coordinate direction i.

The PDEs.jl script includes numerical solvers for the wave equation in n-D using the sparse grid bases: sparse_wave_equation45 and basis sparse_wave_equation78.

A current numerical issue is that there is error generated in the multidimensional coefficients due to limited integration accuracy. This manifests itself more heavily when calculating the derivative, and as a result, many of the error plots are not as nice as we would like, and quickly reach the machine-precision threshold (when p is reasonably large) even for relatively small levels.