Advanced dynamically adaptive algorithms for stochastic simulations on extreme scales

Rick Archibald
Oak Ridge National Laboratory

SIAM Annual Meeting 2011
Numerical Methods for Stochastic Computation and Uncertainty Quantification – Part III of III
Motivation

Given the computed stochastic solutions,

$$f_N(y) = \sum_{i \in I_N} \hat{f}_i \Phi_i(y), \quad y \in \mathbb{R}^d, \quad d \geq 1,$$

the $N$-th degree gPC approximation to an unknown stochastic solution $f(y)$, decompose the domain so the stochastic solution approximation can be computed on smooth sub-regions.
Resources: Hybrid Multi-Core Systems

ORNL supercomputing basics:

- Today: Two Petaflop system with thousands of multicore nodes that uses 6MW.
- 2018: 1000X speed-up in flops (Exaflop), 10x-100x increase in hybrid nodes with only a 3x increase in power

GPU properties

- High flop per watt
- Reliability at scale
- Leverage commodity
- GPU designed to augment CPU
- Complex memory hierarchy
- Boost parallelism
Generalized Edge Detection

**Motivation:** Develop an edge detection method with portability to a broad range of scientific applications.\(^{[1-4]}\)

1. Can be applied to any irregular data in any domain
2. Independent of any specific shape of discontinuities in both the univariate and multivariate case.
3. Easy to implement numerically – cost effective.
4. High order method.

---

Multi-Dimensional Formulation

• Let $\mathcal{S}$ be a set of discrete points in the bounded domain $\Omega \subset \mathbb{R}^d$ and $f$ be a piecewise smooth function known only on $\mathcal{S}$.

• We construct a function, $L_m f$ for $m \in \mathbb{N}$, that has the asymptotical convergence property,

$$L_m f(y) \rightarrow 0,$$

away from jump discontinuities of $f$.

• For any $y \in \Omega$, we choose a set

$$\mathcal{S}_y := S_{m, y} := \{y_1, \ldots, y_{m_d}\},$$

which is a local set of $m_d := \binom{m+d}{d}$ points around $y$. 
The edge detection method is based on a local polynomial annihilation property and performed in the following two steps:

1. Solve the linear system

\[ \sum_{y_j \in S_y} c_j(y) p_i(y_j) = \sum_{|\alpha|_1 = m} p_i(\alpha)(y), \quad \alpha \in \mathbb{Z}^d_+, \quad (1) \]

where \( p_i, i = 1, \cdots, m_d \), is a basis of \( \Pi_m \). Here \( \Pi_m \) denotes the space of all polynomials of degree \( \leq m \) in \( d \in \mathbb{N} \) variables. Note the dimension of \( \Pi_m \) is \( m_d := \binom{m+d}{d} \), and therefore the solution exists and is unique.

2. Our edge detector \( L_m f \) is defined as

\[ L_m f(y) = \frac{1}{q_{m,d}(y)} \sum_{y_j \in S_y} c_j(y) f(y_j). \]

Here \( q_{m,d}(y) \) is a suitable normalization factor depending on \( m \), the dimension \( d \), and the local set \( S_y \).
Minmod Method

The minmod edge detection method was first introduced by Gelb and Tadmor\(^5\) for uniform grids. Here we adopt the same minmod functional, as defined below, to our edge detector at various orders of \(m\).

**Definition**

For a given finite set \(\mathcal{M} \subset \mathbb{N}\) of positive integers, consider the set 
\[
L_M f = \{L_m f : \mathbb{R} \rightarrow \mathbb{R} | m \in \mathcal{M}\}.
\]

The minmod functional is defined by

\[
MM\left(L_M f(y)\right) = \begin{cases} 
  \min_{m \in \mathcal{M}} L_m f(y), & \text{if } L_m f(y) > 0, \forall m \in \mathcal{M}, \\
  \max_{m \in \mathcal{M}} L_m f(y), & \text{if } L_m f(y) < 0, \forall m \in \mathcal{M}, \\
  0, & \text{otherwise.}
\end{cases}
\]

---

\[ f(x) = \begin{cases} 
1, & \sum_{i=1}^{k} x_i^2 < r^2, \ k \leq d \\
0, & \text{otherwise.} 
\end{cases} \]

<table>
<thead>
<tr>
<th>(d)</th>
<th>(N)</th>
<th>(\epsilon_{\text{max}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>573</td>
<td>0.03206</td>
</tr>
<tr>
<td>10</td>
<td>883</td>
<td>0.03160</td>
</tr>
<tr>
<td>15</td>
<td>1193</td>
<td>0.03156</td>
</tr>
<tr>
<td>20</td>
<td>1503</td>
<td>0.03075</td>
</tr>
<tr>
<td>25</td>
<td>1813</td>
<td>0.03055</td>
</tr>
<tr>
<td>50</td>
<td>3363</td>
<td>0.03137</td>
</tr>
<tr>
<td>100</td>
<td>6463</td>
<td>0.03128</td>
</tr>
</tbody>
</table>

Sparse Grids on Hybrid Multi-Core Systems

Can it be done? Yes!??

The sparse grid approximation is based upon the one-dimensional interpolation formula for the univariate function $u^{(1)}$

$$\mathcal{I}_k u^{(1)} = \sum_{j=1}^{m_k} u^{(1)}(x_k^j) \cdot \Psi_k^j(x)$$

(1)

for a set of abscissas $x_k(x_k^1, \ldots, x_k^{m_k})$ and univariate bases $\{\Psi_k^j\}_{j=1}^{m_k}$. Superficially we can approximate $u$ using Smolyak’s algorithm

$$\mathcal{I}_l u = \sum_{l-d+1 \leq |k| \leq l} (-1)^{l-|k|} \binom{d-1}{l-|k|} \cdot (\mathcal{I}_{k_1} \otimes \cdots \otimes \mathcal{I}_{k_N}) u(x)$$

(2)

where $k = (k_1, \ldots, k_d)$ and $|k|_1 = \sum_{j=1}^{d} k_j$. Of all the possible tensor products of one-dimensional formulas only the combinations associated with the indices $l - d + 1 \leq |k|_1 \leq l$ are considered
High Order Hierarchical Basis

- Chebyshev-based node distribution
- Nested quadrature points
- Lagrangian polynomials calculated using stable Barycentric method
Sparse Grid in Two Dimensions and Two Levels
Load balanced distribution of Sparse Grid

Process 0
Grids
00,10,01

Process 1
Grids
20,02

Process 2
Grids
11
Communication for Coefficient Transformation

Process 0
Grids
00,10,01

Process 1
Grids
20,02

Process 2
Grids
11
Communication for Interpolation

Process 0

Grids
00,10,01

Process 1

Grids
20,02

Process 2

Grids
11
Transformation Map: $d=3, n=5$
Transformation Map: $d=3, n=5$
Maximize GPU Performance

- Sufficient parallelism
- Coherent memory access
- Coherent flow control
Fermi Memory Hierarchy Review

SC10 Fundamental Optimizations
Thank You.