An Adaptive Sparse Grids

SuYuzhang

Report in December,2022

Section 3 has been added compared to the November report.

1 Interpolation

The sparse grid method is a method used for interpolation or integration. It selects a subset of the full tensor product collocation points to construct the interpolation. It is an approximation of the full tensor product method, which is a linear combination of tensor product formulas. It sacrifices the accuracy of logarithmic loss and obtains a large computational reduction, which effectively alleviates the curse of dimensionality in the case of high dimensions.

1.1 1D Interpolation

Let $f : [0,1] \to \mathbb{R}$ be a function in 1D, We can approximate it with the following interpolation formula :

$$U^k(f) = \sum_{j=1}^{m_k} f(\xi_j^k) l_j^k(\xi) \ , k \geq 1 \ , U^0 = 0$$

U is the interpolation operator with the set of support nodes $\mathcal{X}^k = \{\xi_j^k \mid \xi_j^k \in [0,1], j = 1, 2..., m_k\}$ and interpolation basis functions $l^k = \{l_j^k \mid l_j^k \in \mathcal{C}[0,1], j = 1, 2, ..., m_k\}$, the interpolation basis functions satisfy $l_j^k(\xi_i^k) = \delta_{ij}$. Here k and m_k refer to the depth of interpolation and the total number of support nodes at depth k, respectively. The chioce of support nodes and interpolation basis functions can be as follows[1],

(1) Equidistant nodes with piecewise linear basis functions

$$m_k = \begin{cases} 1 & if \ k = 1 \\ 2^{k-1} + 1 & if \ k > 1 \end{cases}$$

$$\xi_j^k = \begin{cases} 0.5 & for \ j = 1 \ if \ m_k = 1 \\ \frac{j-1}{m_k - 1} & for \ j = 1, ..., m_k \ if \ m_k > 1 \end{cases}$$

$$l_j^k(\xi) = 1, k = 1$$

$$l_j^k(\xi) = \begin{cases} 1 - (m_k - 1) \mid \xi - \xi_j^k \mid, & if \mid \xi - \xi_j^k \mid < \frac{1}{m_k - 1} \\ 0, & otherwise, \end{cases}$$

(2) Chebyshev Gauss–Lobatto nodes (CGL) with Lagrange polynomial basis functions

$$m_k = \begin{cases} 1 & if\ k = 1 \\ 2^{k-1} + 1 & if\ k > 1 \end{cases}$$

$$\xi_j^k = \begin{cases} 0.5 & for\ j = 1\ if\ m_k = 1 \\ (1 - \cos(\frac{(j-1)\pi}{m_k - 1}))/2 & for\ j = 1, ..., m_k\ if\ m_k > 1 \end{cases}$$

$$l_j^k(\xi) = \begin{cases} 1, & k = 1 \\ \prod_{i=1, i\neq j}^{m_k} \frac{\xi - \xi_i^k}{\xi_j^k - \xi_i^k}, & k > 1, j = 1, ..., m_k \end{cases}$$

It is easy to see that the points are nested, that is $\mathcal{X}^k \subset \mathcal{X}^{k+1}$

1.2 Hierarchical interpolation

Due to the property of nested points the interpolation can be written as a hierarchical form[2]. We first define the incremental interpolant $\Delta^k = U^k - U^{k-1},\ \forall k \geq 1$. Thus,

$$\Delta^k(f) = U^k(f) - U^{k-1}(f)$$

and we have $U^{k-1}(f) = U^k(U^{k-1}(f))$. Using this, we obtain

$$\Delta^k(f) = \sum_{\xi_j^k \in \mathcal{X}^k} f(\xi_j^k) l_j^k - \sum_{\xi_j^k \in \mathcal{X}^k} U^{k-1}(f)(\xi_j^k) l_j^k$$

$$= \sum_{\xi_j^k \in \mathcal{X}^k} (f(\xi_j^k) - U^{k-1}(f)(\xi_j^k)) l_j^k$$

due to $f(\xi_j^k) - U^{k-1}(f)(\xi_j^k) = 0\ \forall\ \xi_j^k \in \mathcal{X}^{k-1}$, we have

$$\Delta^k(f) = \sum_{\xi_j^k \in \mathcal{X}_\Delta^k} (f(\xi_j^k) - U^{k-1}(f)(\xi_j^k)) l_j^k$$

where $\mathcal{X}_\Delta^k$ denotes the nodes added by interpolation from depth $k-1$ to depth k due to the property of nested nodes, we can easy to see it have $m_k^\Delta = m_k - m_{k-1}$ nodes. Thus we can rewrite above formula as,

$$\Delta^k(f) = \sum_{j=1}^{m_k^\Delta} \underbrace{(f(\xi_j^k) - U^{k-1}(f)(\xi_j^k))}_{w_j^k} l_j^k$$

We define w_j^k as the 1D hierarchical surpluses, which is the difference between the actual function value and the value obtained using the interpolant at the $\mathcal{X}_\Delta^k$. The l_j^k we call it hierarchical basis functions. By shifting the terms we can get,

$$U^k(f) = U^{k-1}(f) + \Delta^k(f) = \sum_{i=1}^{k} \Delta^i(f)$$

the set of support nodes can be rewritten as $\mathcal{X}^k = \bigcup_{i=1}^k \mathcal{X}_\Delta^i$.

1.3 Multi-variate interpolation

The multi-variate interpolation formula could simply use tensor product of univariate interpolation formula to construct, given as

$$U^{k_1} \otimes \cdots \otimes U^{k_d}(f) = \sum_{\xi_{j_1}^{k_1} \in \mathcal{X}^{k_1}} \cdots \sum_{\xi_{j_d}^{k_d} \in \mathcal{X}^{k_d}} f(\xi_{j_1}^{k_1}, \ldots, \xi_{j_d}^{k_d}) \cdot (l_{j_1}^{k_1} \otimes \cdots \otimes l_{j_d}^{k_d})$$

where $\mathbf{k} = [k_1, \ldots, k_d]$ represents the depth of interpolation used in each dimension. The hierarchical form can be rewritten as,

$$U^{k_1} \otimes \cdots \otimes U^{k_d}(f) = \sum_{i_1=1}^{k_1} \cdots \sum_{i_d=1}^{k_d} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)$$

2 Sparse Grids

Sparse grid interpolation is a linear combination of tensor products of one-dimensional interpolation. In the high-dimensional case, it obtains a large reduction in computational effort by sacrificing some interpolation accuracy. It is an approximation of the full tensor product method and also known as *Smolyak algorithm*[3], the algorithm give the formula as,

$$A(q,d)(f) = \sum_{|\mathbf{k}| \leq d+q} (\Delta^{k_1} \otimes \cdots \otimes \Delta^{k_d})(f) = A(q-1,d)(f) + \Delta A(q,d)(f)$$

with $A(-1,d) = 0$, and $|\mathbf{k}| = k_1 + \ldots k_d$. The d-dimensional incremental sparse interpolant $\Delta A(q,d)(f)$, can be written as,

$$\begin{aligned}\Delta A(q,d)(f) &= \sum_{|\mathbf{k}|=d+q} (\Delta^{k_1} \otimes \cdots \otimes \Delta^{k_d})(f) \\ &= \sum_{|\mathbf{k}|=d+q} \sum_{\mathbf{j}} \underbrace{(l_{j_1}^{k_1} \otimes \cdots \otimes l_{j_d}^{k_d})}_{l_{\mathbf{j}}^{\mathbf{k}}} \cdot \underbrace{(f(\xi_{j_1}^{k_1}, \ldots, \xi_{j_d}^{k_d}) - A(q-1,d)(f)(\xi_{j_1}^{k_1}, \ldots, \xi_{j_d}^{k_d}))}_{w_{\mathbf{j}}^{\mathbf{k}}}\end{aligned}$$

where $\mathbf{j} = (j_1, \ldots, j_d)$ denotes the multi-index. As for the 1D case, the coefficients $w_{\mathbf{j}}^{\mathbf{k}}$ are defined as *hierarchical surpluses*. Most of the time people use its explicit form,

$$A(q,d) = \sum_{q-d+1 \leq |\mathbf{k}| \leq q} (-1)^{q-|\mathbf{k}|} \binom{d-1}{q-|\mathbf{k}|} \cdot (U^{k_1} \otimes \cdots \otimes U^{k_d})(f)$$

3 Adaptive Sparse Grids

For sufficiently smooth functions, the sparse grid method can effectively reduce the amount of computation. However, for functions with discontinuities or large slopes, the sparse grid method will converge slowly or even fail to converge. And in general calculations, we do not know whether the function we are solving for has this discontinuity property or not. Therefore we need to use an adaptive sparse grid to deal with this class of problems. The current popular adaptive approach is introduced by Ma and Zabaras[4]. The method uses equidistant nodes as well as piecewise multi-linear basis functions to construct the interpolation. The choice of nodes and basis functions is based on their local support, and this method does not have any explicit decomposition for random domains. This report focuses on a domain adaptive sparse grid and will introduce its application in the stochastic collocation approach[1].

3.1 Adaptive criterion

Interpolation domain (called random domain in UQ) was characterized by n mutually independent random variables $\boldsymbol{\xi} = [\xi_i]_{i=1}^n$, where $\boldsymbol{\xi} : \Theta \to \Gamma = [0,1]^n$. As the adaptive refinement procedure proceeds, the random domain Γ is decomposed into N_d non-overlapping subdomains,such that $\Gamma = \bigcup_{s=1}^{N_d} \Gamma^s$.Define an indicator function $I_s(\xi)$ on each subdomain s,

$I_s(\xi) = \begin{cases} 1 & if\ \boldsymbol{\xi} \in \Gamma^s, \\ 0 & otherwise. \end{cases}$

use this function,we can define our first criterion,

$\beta_s J_s \geq \tau_1$,

where,

$\beta_s = \max\limits_{|\mathbf{k}|=d+q_s} \left(\left| \mathbf{w}_{\mathbf{j}}^{\mathbf{k}} \right| \right)$

Which is the maximum value of the hierarchical surpluses of nodes newly added in all directions, $J_s = Pr(I_s = 1)$ is the probability that ξ lies in sth-subdomain, and τ_1 is the prescribed error tolerance. $\mathbf{k} = [k_1, \ldots, k_d]$ represents the depth of interpolation used in each dimension,and $\mathbf{j} = [j_1, \ldots, j_d]$ indicates the serial number of the collocation nodes in each direction, such that $j_e = 1, 2, ..., m_{k_e}^{\Delta}$,and $e = 1, 2, ..., n$.

The second criterion is

$\gamma_i \geq \tau_2 \cdot (\max\limits_{j=1,\ldots,n} \gamma_j),\ 0 < \tau_2 < 1, i = 1, ..., n$

where,

$\gamma_i = \sum\limits_{\mathbf{j},\mathbf{k}=\mathbf{k}_i} (\mathbf{w}_{\mathbf{j}}^{\mathbf{k}})^2,\ i = 1, ..., n,\ \mathbf{k}_i = \{\mathbf{k} : k_i = q_s + 1, k_j = 1\ \forall\ j \neq i\}$

The first criterion determines whether a random domain s is divided and second criterion determines which direction to divide. After refinement, the random domain is broken down into a number of sub-domains on which we perform calculations, and the results of each sub-domain are combined to give us our final result.

3.2 Computation of moments

In the UQ problem, we need to calculate the statistical moments of order m.Once we have obtained the approximation in the local sub-domain, we can assemble it to obtain the global statistical moments as described by Wan and Karniadakis in [5] . Conditional PDF given as,

$$\rho^s(\boldsymbol{\eta}^s) = \frac{\rho(\boldsymbol{\eta}^s)}{Pr(I_s = 1)}$$

where $\rho^s(\boldsymbol{\eta}^s)$ is the global PDF and $\rho(\boldsymbol{\eta}^s)$ is local. $\boldsymbol{\eta}^s : I_s^{-1}(1) \to \Gamma^s$ is a random vector. Using law of total probability, mth-moment of the stochastic solution can given as,

$$\langle u^m(\boldsymbol{\xi})\rangle \approx \int_\Gamma \hat{u}^m(\boldsymbol{\xi})\rho(\boldsymbol{\xi})d\boldsymbol{\xi} = \sum_{s=1}^{N_d} Pr(I_s = 1) \int_{\Gamma^s} \hat{u}^m(\boldsymbol{\eta}^s)\rho^s(\boldsymbol{\eta}^s)d\boldsymbol{\eta}^s$$

We use another random vector $\boldsymbol{\xi} = g_s(\boldsymbol{\eta}^s) : I_s^{-1}(1) \to [0,1]^n$ to map the Γ^s to $[0,1]^n$,where,

$$g_s(\boldsymbol{\eta}^s) : \eta_i^s = (b_i^s - a_i^s)\xi_i^s + a_i^s,\ i = 1, ..., n,$$

and $(a_i^s, b_i^s), i = 1, ..., n$ define the subdomain $\Gamma^s = [a_1^s, b_1^s) \times [a_2^s, b_2^s) ... \times [a_n^s, b_n^s)$.Thus the formula can rewritten as,

$$\langle u^m(\boldsymbol{\xi})\rangle \approx \sum_{s=1}^{N_d} Pr(I_s = 1) \int_{[0,1]^n} \hat{u}_s^m(\boldsymbol{\xi}^s)\rho^s(\boldsymbol{\xi}^s)d\boldsymbol{\xi}^s$$

where $\hat{u}_s$ represents the approximate solution in sth-subdomain.

References

[1] N. Agarwal and N. R. Aluru, "A domain adaptive stochastic collocation approach for analysis of mems under uncertainties," *Journal of Computational Physics*, vol. 228, no. 20, pp. 7662–7688, 2009.

[2] A. Klimke, "Piecewise multilinear sparse grid interpolation in matlab," 2003.

[3] S. A. Smolyak, "Quadrature and interpolation formulas for tensor products of certain classes of functions," in *Doklady Akademii Nauk*, vol. 148, no. 5. Russian Academy of Sciences, 1963, pp. 1042–1045.

[4] X. Ma and N. Zabaras, "An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations," *Journal of Computational Physics*, vol. 228, no. 8, pp. 3084–3113, 2009.

[5] X. Wan and G. E. Karniadakis, "An adaptive multi-element generalized polynomial chaos method for stochastic differential equations," *Journal of Computational Physics*, vol. 209, no. 2, pp. 617–642, 2005.