Spring 2019: Advanced Topics in Numerical Analysis: Stochastic modeling and uncertainty quantification in complex systems Assignment 2 (due Apr. 22, 2019)

Terrence Alsup

1. Smolyak quadrature

We implement the Smolyak quadrature by creating a sequence of helper functions. The first is `cheb1D`, which takes in an integer and computes the Clenshaw-Curtis points and weights in 1-dimension. We use an alternate formula for the points, which is actually the same one that the package Chebfun uses.

$$x_{l,k} = \sin\left(\frac{\pi k}{2(n_l - 1)}\right) \quad \text{for } k = -(n-1), -(n-1)+2, \ldots, n-1 \tag{1}$$

This formula enforces a symmetry so that at the middle point we compute 0 exactly and obtain better accuracy. Next in our implementation we have a function called `chebdD`, which computes the tensor-product of the Clenshaw-Curtis points for multiple dimensions and calculates the weights by taking the product. After this we have `incdD` which computes the points and weights for the increments (again tensorized). Both this function and `chebdD` are recursive in the dimension when computing the tensor-product of the points. Finally we implement `smolyakdD`, which takes as inputs the level, the dimension, and a function handle. We use a helper function called `integer_combinations` that computes all of the possible integer combinations of d variables to achieve a total value. This is done in a recursive fashion on the dimension again. Finally, to compute the integral of the function with respect to the quadrature rule we sum over all of the results from the increments.

We now test our implementation of the Smolyak quadrature on computing the integral

$$\int_{[0,1]^d} (1+1/d)^d \prod_{i=1}^{d} x_i^{(1/d)} d\boldsymbol{x} = 1 \tag{2}$$

by looking at the relative error as in Table 1. We see two things. First is that as we increase the level the relative error of our approximation decreases, indicating that the implementation is working correctly. The second is that as we increase the dimension the relative error decays much more slowly. Indeed for $d = 6$ we still have a fairly large relative error even at level 10.

Level $\backslash d$	1	2	3	4	5	6
4	0.006448	0.121020	0.357042	0.595544	0.771527	0.880616
6	0.000402	0.022756	0.121988	0.297871	0.497392	0.672269
8	0.000025	0.003796	0.033994	0.119038	0.259617	0.428249
10	0.000002	0.000593	0.008376	0.040617	0.114351	0.231345

Table 1: The relative error of our Smolyak quadrature rule for the integral 2.

We also test our implementation on computing the integral

$$\int_{[-1,1]^d} \prod_{i=1}^{d} \sin\left(\frac{(x_i+1)\pi}{2}\right) d\boldsymbol{x} = \left(\frac{4}{\pi}\right)^d \tag{3}$$

and have recorded the relative error in Table 2 below. We again see that increasing the level greatly improves the accuracy, but increasing the dimension results in slower convergence. Additionally, we notice that these relative errors are much smaller and converge to zero much faster than when calculating 2. This agrees with the theory behind sparse grids because the function we are integrating in 3 is much smoother. Thus, we would expect faster convergence.

Level $\backslash d$	1	2	3	4	5	6
1	0.047198	0.096623	0.148381	0.202581	0.259340	0.318778
2	0.000296	0.001607	0.005918	0.012860	0.022672	0.035609
3	0.000000	0.000028	0.000018	0.000252	0.000807	0.001832
4	0.000000	0.000000	0.000002	0.000003	0.000005	0.000037

Table 2: The relative error of our Smolyak quadrature rule for the integral 2.

2. Stochastic collocation

We now test our implementation on the model problem with $\boldsymbol{\xi} = [\xi_1, \ldots, \xi_d]$ and $\xi_i \sim \text{Uniform}[1, 10]$ are i.i.d. We write $Q(u; \boldsymbol{\xi})$ to denote our quantity of interest which depends implicitly on $\boldsymbol{\xi}$, and we calculate the expectation by

$$\mathbb{E}_{\boldsymbol{\xi}}[Q(u)] = \frac{1}{9^d} \int_{[1,10]^d} Q(u; \boldsymbol{\xi})\, d\boldsymbol{\xi}. \tag{4}$$

To fit with our implementation, which can only integrate over $[-1, 1]^d$, we use the change of variables $x_i = \frac{2}{9}\xi_i - \frac{11}{9}$. Now the integral becomes

$$\mathbb{E}_{\boldsymbol{\xi}}[Q(u)] = \frac{1}{2^d} \int_{[-1,1]^d} Q(u; \boldsymbol{x})\, d\boldsymbol{x}. \tag{5}$$

We compare the Smolyak quadrature with the Monte Carlo estimator

$$\mathbb{E}_{\boldsymbol{\xi}}[Q(u)] \approx \frac{1}{N} \sum_{i=1}^{N} Q(u^{(i)}; \boldsymbol{\xi}^{(i)}), \qquad \boldsymbol{\xi}^{(i)} \text{ i.i.d.} \tag{6}$$

For $d = 2$ we split the domain evenly into a top half $\{(x_1, x_2) : x_2 \in (0.5,\ 1]\}$ and a bottom half $\{(x_1, x_2) : x_2 \in [0,\ 0.5]\}$ in the function `coeffFun2DBlocks2`. Since we are in 2 dimensions, we can calculate a reference value Q_{ref} using Matlab's built-in `integral2` function. Figure 1 below shows the comparison between Smolyak quadrature with the Monte Carlo estimator 6. We see that Smolyak quadrature performs significantly better than Monte Carlo and has an extremely fast rate of decay for the error. This is a result of our quantity of interest $Q(u, \boldsymbol{\xi})$ depending smoothly on the parameter $\boldsymbol{\xi}$. Monte Carlo has no way to take advantage of this and is why we observe the usual rate of $1/2$.

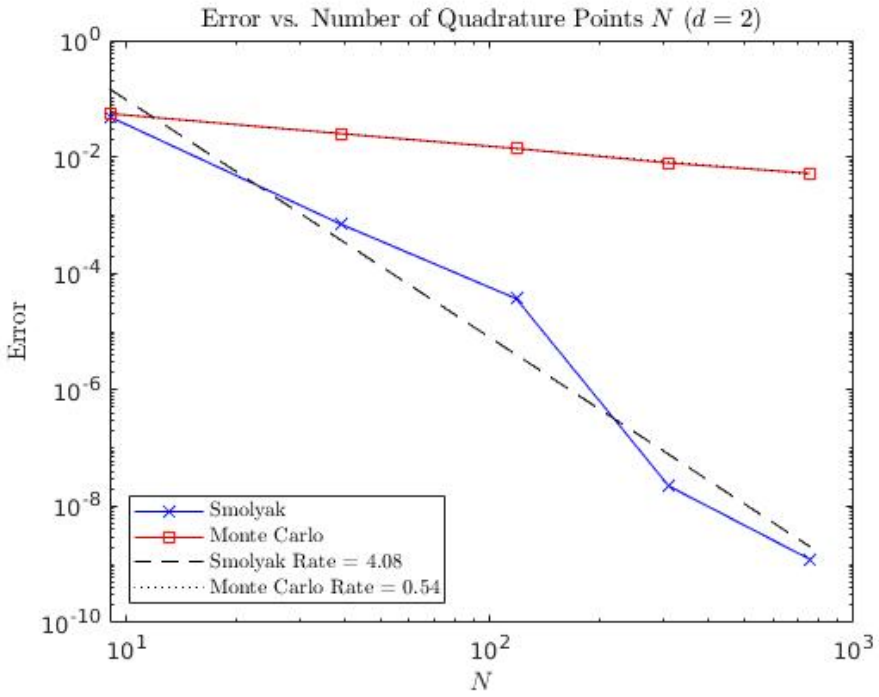

Figure 1: The rate of convergence for Smolyak quadrature and Monte Carlo for computing the integral 4 when $d = 2$. Note that the solution $Q(u)$ itself is approximated with finite elements using level $\ell = 4$. For Monte Carlo we estimate the root-mean-squared-error (RMSE) using 100 trials.

Similarly, we can increase the dimension to $d = 6$ by splitting the domain into 6 regions

$$\Omega_i = \left\{(x_1, x_2) \ : \ \frac{1}{6}(i-1) \leq x_2 < \frac{1}{6}i\right\} \quad i = 1, 2, \ldots, 6, \tag{7}$$

which is done with the Matlab function `coeffFun2DBlocks6`. Since there is no built-in Matlab function to compute this 6-dimensional integral we calculate a reference value Q_{ref} by using the Smolyak quadrature (now that it has been tested) with sparse grids of level 4 (note this is different from the level $\ell = 2$ used to solve the PDE). Figure 2 below shows the results for sparse grid levels $1, 2, 3$. As expected Smolyak quadrature again performs better than Monte Carlo although this time the rate is slower. This is a result of solving a higher-dimensional integral. It is also potentially the result of less regularity of $Q(u; \boldsymbol{\xi})$. We note that here we used the level $\ell = 2$ when solving for $Q(u)$ whereas before we had used $\ell = 4$. This was an attempt to greatly reduce computational cost.

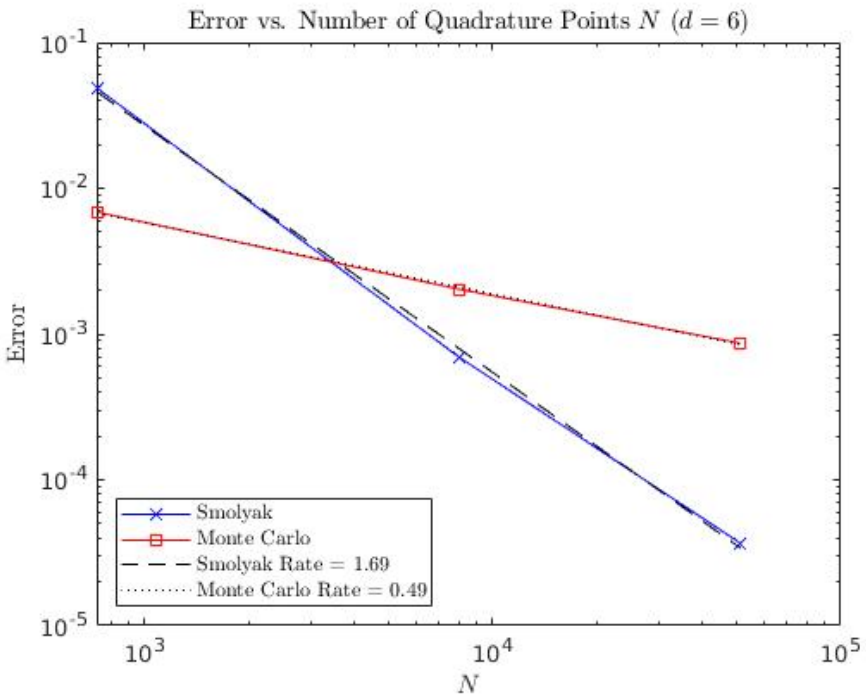

Figure 2: The rate of convergence for Smolyak quadrature and Monte Carlo for computing the integral 4 when $d = 6$. Note that the solution $Q(u)$ itself is approximated with finite elements using level $\ell = 2$. For Monte Carlo we estimate the root-mean-squared-error (RMSE) using 100 trials.

We note that an attempt to go to higher dimensions, particularly $d = 16$, fails even for the sparse grids. The base level has 3 quadrature points in each dimension. For 16 dimensions this is 3^{16} total quadrature points, which is over 43 million. This is way too expensive for either Smolyak or Monte Carlo given that each point requires one solve of our PDE, even with parallelization. Note that for Monte Carlo we could choose to use fewer samples, but would then get a fairly large RMSE.