

Solving Stochastic Optimization by Newton-type methods with Dimension-Adaptive Sparse Grid Quadrature

Yuancheng Zhou

February 22, 2022

Abstract

Stochastic optimisation problems minimise expectations of random cost functions. We use 'optimise then discretise' method to solve stochastic optimisation. In our approach, accurate quadrature methods are required to calculate the objective, gradient or Hessian which are in fact integrals. We apply the dimension-adaptive sparse grid quadrature to approximate these integrals when the problem is high dimensional. Dimension-adaptive sparse grid quadrature shows high accuracy and efficiency in computing an integral with a smooth integrand. It is a kind of generalisation of the classical sparse grid method, which refines different dimensions according to their importance. We show that the dimension-adaptive sparse grid quadrature has better performance in the 'optimise then discretise' method than the 'discretise then optimise' method.

1 Introduction

Stochastic optimisation is a useful tool in decision making and has many applications. The general form of an unconstrained stochastic optimisation problem is

$$\min_{u \in U} \mathbb{E}[h(u, W)], \quad (1)$$

where W is a d dimensional random vector which is defined on the probability space $(\Omega, \mathcal{B}, \mathbb{P})$, \mathcal{B} is the Borel σ -algebra and \mathbb{P} is the associate probability measure. U is a subset of \mathbb{R}^n which contains all possible decisions.

If the random vector W subjects to a probability density $p(w)$ on \mathbb{R}^d the objective is of the form

$$F(u) := \mathbb{E}[h(u, W)] = \int_{\mathbb{R}^d} h(u, w)p(w) dw = \int_{\mathbb{R}^d} f(u, w) dw \quad (2)$$

where $f(u, w) = h(u, w)p(w)$.

There are two categories of approaches to solve the stochastic optimisation problem. One is based on the idea of 'discretise then optimise' while the other is based on 'optimise then discretise'. When we apply the 'discretise then optimise' method(DTOM) to solve the stochastic optimisation problem, it turns out to be

the so called scenario generation method. The main idea of this kind of methods is to first approximate the integrals (2) by a quadrature rule Q ,

$$F(u) \approx S(u) = Q(f(u, \cdot)). \quad (3)$$

then use $S(u)$ as a surrogate objective function and minimise it. The key of the scenario generation method is to find a good approximation to the original objective function. Monte Carlo(MC) and Quasi Monte Carlo(QMC) methods are successfully used in scenario generation for many applications. There are also many studies in convergence of the MC and QMC methods [8, 23, 18, 19, 24]. Recently, Michael, Sanjay and David developed the scenario generation via the sparse grid method (SGSG) [3, 4]. The advantage of their approach, as shown in their paper, is the SGSG method converges faster than these scenario generation methods based on MC and QMC if the integrand in the objective function is smooth enough. They also showed the epi-convergence of the SGSG. However, there are also disadvantages of SGSG. When an integral is approximated using a sparse grid quadrature, weights of some grid points can be negative. The problem brought from these negative weights is that some important properties of the objective function, e.g. convexity, are no longer kept [26]. The original convex objective function can be replaced by a non-convex surrogate function in SGSG method. This will bring more difficulties in computing the problem, e.g. from convex optimisation to non-convex optimisation, and analyse the performance of the whole algorithm.

Here we will study the alternative 'optimise then discretise' method (OTDM) and apply the idea to solve the stochastic optimisation problems. The basic idea of our approach is to solve the system of equations

$$\nabla F(u) = G(u) = 0. \quad (4)$$

There are many numerical methods can be used to solve the system of the non-linear equations. In order not to be too general, we focus on the Newton-type methods in this paper. For the integrals appear during the computation of the gradient G , we will apply more sophisticated dimension adaptive sparse grid method.

Newton's method and its variants have been widely used in solving nonlinear optimisation problems and nonlinear equations [16, 17]. Hessian matrix needs to be computed in the original Newton's method while most its variants such BFGS and L-BFGS-B are gradient based method. The iterations of Newton-type method can be written into the following form

$$u_{p+1} = u_p - A_p^{-1}G(u_p), \quad p = 1, 2, \dots \quad (5)$$

where A_p is an approximation to the Hessian Matrix. The convergence of the Newton-type methods are well studied [16]. Newton's method is quadratically convergent when the initial value is close enough to the minimiser. Other Newton-type methods(Quasi Newton methods) have lower convergence rate than Newton's method [16], however, they are used more frequently in practical computation since the Hessian matrix is not required to be computed and stored. These convergence theories can make sure the sequence $\{u_p\}$ generated by the (5) converges to a minimiser. However, in practice, one can only get a perturbed sequence $\{\bar{u}_p\}$ rather than the ideal sequence u_p . This is because of

the existence of both rounding errors and truncated errors during the approximation to the function value F , gradient G and the computation of iterations (5). It is shown in [25, 16, 7] that if the error from an approximation of the gradient G is sufficiently small, then the perturbed Newton-type method will produce a sequence $\{\bar{u}_p\}$ which converges to a minimiser u^* . This result also implies the Newton-type method can stop convergence if the error in the approximation of the gradient G is large to some extent. However, computing the gradient G involves computing high dimensional integrals when d is large. It is difficult to get a very accurate approximation in this case. Thus it is important for us to know when to stop the iterations in the Newton-type method. If we stop it too early, the solution can not achieve its best potential accuracy. If we stop too late, we will waste a great amount of computational cost. We offer the stopping criterion for our method based on the error analysis mentioned in [25].

If we assume the integral and the derivative are interchangeable, then the gradient G is

$$G(u) = \nabla F(u) = \int_{\Omega} \nabla f(u, w) dw. \quad (6)$$

Each component of G is an integral. As we have mentioned before, if these integrals are high dimensional, it will be very difficult to compute them even for moderate accuracy. This is so called 'curse of dimensionality'. However, for special function classes, such as functions which have bounded mixed derivatives, the sparse grid method [11, 2, 9] can mitigate the curse of dimensionality to a large extent. The performance of the sparse grid method can be further improved if we treat each dimension differently. Actually, in many applications, the importance of different dimensions are not equal. This property inspires the idea of the dimension-adaptive sparse grid [14, 10, 13, 12, 15]. Unlike the classical sparse grid method which treats all the dimensions equally during the computation, the dimensional adaptive sparse grid method always refines the most 'important' dimension first. The performance is thus improved for those integrals with dimensions of different importance. Both the sparse grid method and the dimension-adaptive sparse grid method are used when we compute high dimensional integrals in the computation.

This paper is organised as follows. In section 2, we introduce basic concepts and results in optimisation which we need to use. In section 3, we review the sparse grid and dimension adaptive sparse grid method. The 'optimise then discretise' method and algorithms based on it are given in section 4. In section 5, we study the convergence of the sequence generated by the algorithm in section 4. In section 6, we develop the stopping criterion for our method. Finally, we show some high dimensional examples in the last section.

2 Basic concepts of optimisation

First we begin with introducing some basic concepts of optimisation.

Definition 2.1. u^* is a global minimizer of F if $F(u) \geq F(u^*)$ for all $u \in \mathbb{R}^n$.

Definition 2.2. F is convex when F satisfies

$$F(tu + (1-t)v) \leq tF(u) + (1-t)F(v), \quad \forall u, v \in \mathbb{R}^n, t \in [0, 1].$$

F is γ -strongly convex when there exists $\gamma > 0$ such that

$$F(tu + (1-t)v) \leq tF(u) + (1-t)F(v) - \frac{1}{2}\gamma t(1-t)\|u-v\|_2^2, \quad \forall u, v \in \mathbb{R}^n, t \in [0, 1].$$

We then make some smoothness assumptions on the cost function F such that the global minimiser of the stochastic optimisation problem exists.

Definition 2.3. A function $g : \mathbb{R}^p \rightarrow \mathbb{R}^q$ is Lipschitz continuous with constant $L > 0$ if

$$\|g(x) - g(y)\|_2 \leq L\|x - y\|_2, \quad \forall x, y \in \mathbb{R}^q.$$

Assumption 2.1. The function F is continuously differentiable and ∇F is Lipschitz continuous with constant $L > 0$. In this case, we call F L -smooth.

By using Taylor expansion, we have $\|\nabla^2 F(u)\|_2 \leq L$ if F is L -smooth and twice continuously differentiable.

The following Lemma gives connection between convexity of a function and its smoothness.

Lemma 2.1 ([20]). If F is continuously differentiable, then F is convex if and only if F lies on or above any tangent line:

$$F(v) \geq F(u) + \nabla F(u)^T(v - u), \quad \forall u, v \in \mathbb{R}^n.$$

Also, F is γ -strongly convex if and only if

$$F(v) \geq F(u) + \nabla F(u)^T(v - u) + \frac{\gamma}{2}\|v - u\|_2^2, \quad \forall u, v \in \mathbb{R}^n.$$

If F is twice continuously differentiable, then F is convex if and only if $\nabla^2 F(w)$ is positive semidefinite for every $w \in \mathbb{R}^n$. Also, F is γ -strongly convex if and only if $\nabla^2 F(w) \geq \gamma I$.

By using this Lemma, we can show the existence and uniqueness of global minimizers for strongly convex functions.

Theorem 2.1 ([20]). If $F : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable and strongly convex, then it has a unique global minimizer.

If the function f satisfies assumptions in Theorem 2.1, we can make sure the stochastic optimisation problem is well defined. Next, we consider the numerical solvers to solve the problem. In order not to be too general, we will use Newton-type methods as our solvers. According to the optimality condition, solving the stochastic optimisation problem is equivalent to solve the following system of equations

$$G(u) = \nabla F(u) = 0. \tag{7}$$

The Newton-type methods generate following sequence $\{u_p\}$

$$u_{p+1} = u_p - A_p^{-1}G(u_p), \quad p = 0, 1, 2, \dots \tag{8}$$

and one expects the limit of this sequence will be the solution of (7). In the iteration, $A_p \in \mathcal{L}(U)$ is an approximation to the derivative $G'(u)$, namely, the

Hessian of F . A_p can be generated in many different ways and different choices of A_p lead to different kinds of Newton-type methods. For example, if we take

$$A_p = \nabla^2 F(u_p),$$

this is exactly Newton's method. If we take

$$A_p = \alpha_p^{-1} B_p,$$

where α_p is chosen by exact/inexact line search and B_p is updated from the previously computed value B_{p-1}

$$B_p = B_{p-1} + \frac{y_{p-1} y_{p-1}^T}{s_{p-1}^T y_{p-1}} - \frac{(B_{p-1} s_{p-1})(B_{p-1} s_{p-1})^T}{s_{p-1}^T B_{p-1} s_{p-1}},$$

where $B_0 = I$, $s_p := u_{p+1} - u_p$ and $y_p := \nabla F(u_{p+1}) - \nabla F(u_p)$, then the iteration (8) becomes the BFGS method [16, 17], one of the most frequently used Quasi Newton methods.

If we further assume $\nabla^2 F(u)$ is definite, bounded and Lipschitz continuous, see details in [16, 17], the Newton method is quadratically convergent when the initial value is close enough to the minimiser while the BFGS method is superlinearly convergent. We can similarly assume the positive definiteness and boundedness of $\nabla^2 f(u, w)$, $\forall w \in \Omega$ and the Lipschitz continuity of $\nabla^2 f(u, w)$, $\forall w \in \Omega$ to make sure Newton method and BFGS method are convergent.

When we consider high dimensional problems, the difficulty lies in the approximations of $F(u)$, $\nabla F(u)$ and $\nabla^2 F(u)$. The objective $F(u)$ and each component of $\nabla F(u)$ and $\nabla^2 F(u)$ are high dimensional integrals in such case. Moreover, we have to compute them at each iteration in the solvers. This will result in the curse of dimensionality.

3 Dimension-adaptive sparse grid

3.1 Formulation of the Dimension-adaptive sparse grid

Here we introduce the sparse grid and the dimension-adaptive sparse grid method to approximate the d dimensional integral

$$\int_{[-1,1]^d} f(x) dx. \quad (9)$$

The sparse grid quadrature is built upon 1D quadrature. Suppose we have a sequence of 1D quadrature rules

$$Q_i(f) = \sum_{x_{i,j} \in G_i} c_{i,j} f(x_{i,j}) \quad (10)$$

where G_i is a set which contains all quadrature points $\{x_{i,j}, j = 1, \dots, N_i\}$ of the j th 1D quadrature rule. N_i is the number of the quadrature points. $\{w_{i,j}, j = 1, \dots, N_i\}$ are the corresponding weights of the j th rule. In particular, we focus on hierarchical quadrature rules here. The hierarchical means the sets G_i are nested, i.e. $G_i \subset G_{i+1}$, $\forall i$.

Once we have a sequence of 1D quadrature rules, we can define the differences

$$\Delta_i(f) = Q_i(f) - Q_{i-1}(f), \quad i = 1, 2, \dots \quad (11)$$

with $Q_0(f) = 0$. By using these differences, the l th quadrature rule can be written as

$$Q_l(f) = \sum_{i=1}^l \Delta_i(f). \quad (12)$$

Multidimensional quadrature rules can be constructed based on the similar idea. Product rule is one of the methods used to deal with multidimensional integrals. The d -dimensional product rule is of the form

$$Q_{\mathbf{l}}(f) = \sum_{x_{\mathbf{l},\mathbf{j}} \in G_{\mathbf{l}}} c_{\mathbf{l},\mathbf{j}} f(X_{\mathbf{l},\mathbf{j}}) \quad (13)$$

where

$$\begin{aligned} G_{\mathbf{l}} &= G_{l_1} \times \dots \times G_{l_d} \\ c_{\mathbf{l},\mathbf{j}} &= c_{l_1,j_1} \dots c_{l_d,j_d} \\ x_{\mathbf{l},\mathbf{j}} &= (x_{l_1,j_1}, \dots, x_{l_d,j_d}). \end{aligned} \quad (14)$$

If we defined the d -dimensional differences as the tensor product of the 1D differences

$$\Delta_{\mathbf{i}}(f) = \Delta_{i_1} \otimes \dots \otimes \Delta_{i_d}(f), \quad (15)$$

then the product rule (13) can be written as

$$Q_{\mathbf{l}}(f) = \sum_{\mathbf{i} \leq \mathbf{l}} \Delta_{\mathbf{i}}(f). \quad (16)$$

Instead of using all the differences in the index set $\{\mathbf{j}, \mathbf{j} \leq \mathbf{l}\}$, the sparse grid quadrature only sums over a subset of it. The level l sparse grid quadrature is then defined by

$$Q_l^d(f) = \sum_{\mathbf{i} \leq l+d-1} \Delta_{\mathbf{i}}(f). \quad (17)$$

In fact, we can build a multidimensional quadrature by summing up any downset \mathbf{I} of the full grid index set $\{\mathbf{i}, \mathbf{i} \leq \mathbf{l}\}$. The downset is defined as below.

Definition 3.1. We say \mathbf{I} is a downset if it satisfies

$$\mathbf{i} \in \mathbf{I} \quad \text{and} \quad \mathbf{m} \leq \mathbf{i}, \quad (18)$$

then $\mathbf{m} \in \mathbf{I}$.

We call the multidimensional quadrature defined on such downset as generalised sparse grid quadrature and

$$Q_{\mathbf{I}}^d(f) = \sum_{\mathbf{i} \in \mathbf{I}} \Delta_{\mathbf{i}}(f). \quad (19)$$

For the generalised sparse grid quadrature, the downset \mathbf{I} is chosen before we compute the quadrature. Different downsets \mathbf{I} are chosen in different applications. The truncated sparse grid, the sparse grid with fault, etc are several frequently used generalised sparse grid.

The dimension-adaptive sparse grid is still with the form

$$Q_{\mathbf{I}}^d(f) = \sum_{\mathbf{i} \in \mathbf{I}} \Delta_{\mathbf{i}}(f). \quad (20)$$

while the downset \mathbf{I} is decided during the computation according to the 'importance' of each dimension. There are two important things needed to be considered before designing the dimension adaptive algorithm. First, when we add a new surplus $\Delta_{\mathbf{i}}f$ to the sum, we need to make sure the newly generated index set $\mathbf{I} \cup \{\mathbf{i}\}$ is still a downset. This is because we need to use the method of differences to compute the telescope sum and thus every index \mathbf{j} which have smaller entries than \mathbf{i} in at least one dimension must be included in \mathbf{I} . Second, the algorithm is required to detect the 'important dimension' and do refinement first in 'the most important dimension' at each iteration.

Algorithm 1 Dimension adaptive sparse grid quadrature

```

Initialize  $\mathbf{I} = \{\mathbf{1}\}$  and  $s = \Delta_{\mathbf{i}}f$ 
while Termination condition not reached do
    Consider all possible covering elements to  $\mathbf{I}$  and put them in a heap  $\mathcal{A}$ 
    Select  $\mathbf{i}$  from heap  $\mathcal{A}$  with largest  $\Delta_{\mathbf{i}}f$ 
     $s = s + \Delta_{\mathbf{i}}f$ 
end while

```

The termination condition we considered in this paper is

$$\mathbf{I} = \{\mathbf{i} \leq \mathbf{l} \mid |\Delta_{\mathbf{i}}f| \geq \epsilon\}. \quad (21)$$

The termination condition $|\Delta_{\mathbf{i}}f| \geq \epsilon$ has been used in many dimension adaptive sparse grid algorithms to stop the while loop [13, 12, 15]. The additional condition $\mathbf{i} \leq \mathbf{l}$ we added here is aimed at avoiding excessive refinement in some dimensions. Here, we say $\{\mathbf{i}\}$ is a covering element of a downset \mathbf{I} if $\mathbf{I} \cup \{\mathbf{i}\}$ is also a downset.

We use $\mathcal{Q}_{\mathbf{l},\epsilon}$ as the operator for the dimension-adaptive sparse grid quadrature in Algorithm 1 with the termination condition (21). The choice of the downset \mathbf{I} depends on f, \mathbf{l}, ϵ , so we have

$$\mathcal{Q}_{\mathbf{l},\epsilon}(f) = Q_{\mathbf{I}(f,\mathbf{l},\epsilon)}(f). \quad (22)$$

It should be noted here we have to use different notations for the quadrature method($\mathcal{Q}_{\mathbf{l},\epsilon}$) and the computing formula($Q_{\mathbf{I}}$). This is because when the same quadrature method applied to approximate different integrals, e.g. integrals with integrand f and g , respectively, we can get different downsets \mathbf{I}_f and \mathbf{I}_g ,

$$\mathbf{I}_f = \mathbf{I}(f, \mathbf{l}, \epsilon) \neq \mathbf{I}(g, \mathbf{l}, \epsilon) = \mathbf{I}_g. \quad (23)$$

Thus, the formula used to approximate the integrals are different, that is,

$$Q_{\mathbf{I}_f} \neq Q_{\mathbf{I}_g}. \quad (24)$$

For non-adaptive approach, we don't have such problem. We use the same notation(Q) to denote both quadrature method and computing formula.

3.2 1D quadrature rules

We use the following three types of 1D quadrature rules to build our dimension-adaptive sparse grid quadratures. They are the trapezoidal rule, the Clenshaw-Curtis rule [5] and the Gauss-Patterson rule [22]. All of these are hierarchical quadrature rules. The trapezoidal rule has $O(4^{-l})$ accuracy on the uniform grid with $N = 2^{l-1} + 1$ grid points when the integrand $f \in C^2$. The accuracy can be further improved to $O(2^{-lr})$ if the integrand $f \in C^r$ is periodic. The $N = 2^{l-1} + 1$ points Clenshaw-Curtis rule uses extremal points of the Chebyshev polynomial of degree $N+1$ as its quadrature points. The $N+1$ points Clenshaw-Curtis rule integrates polynomials of degree less or equal than N exactly [11, 15]. The accuracy of an N points rule is $O(2^{-lr})$ [6, 11]. The Gauss-Patterson rule is a Kronrod extension of the corresponding Gauss rule. The polynomial degree of exactness of an $N = 2^l - 1$ points rule is $(3N - 1)/2$. Its accuracy for the integrals with integrand $f \in C^r$ is also $O(2^{-lr})$ [6, 11] for an N points rule. It is noteworthy that both level l trapezoidal rule and Clenshaw-Curtis rule have $2^{l-1} + 1$ quadrature points while the level l Gauss-Patterson rule has $2^l - 1$ quadrature points.

3.3 Accuracy of high dimensional quadrature rules

We will mainly discuss the accuracy of the dimension-adaptive sparse grid quadrature rules. Before that, we first provide the results on product rule and sparse grid quadrature.

The computational complexity of the product rule is $O(N_l^d)$ for $l_i = l$. However, the accuracy is $O(2^{-lr})$. Here we notice that the accuracy is not depend on the dimension which results in the curse of dimensionality.

For the sparse grid quadrature rules, If we assume the integrand f has bounded mixed derivatives up to order r , i.e., $f \in H^r([-1, 1]^d)$ where

$$H^r([-1, 1]^d) = \left\{ f : [-1, 1]^d \rightarrow \mathbb{R} : \max_{|\alpha|_\infty \leq r} \left\| \frac{\partial^{|\alpha|_1} f}{\partial \alpha W} \right\| < \infty \right\} \quad (25)$$

where $\|\cdot\|$ denotes the L_2 norm and $|\alpha|_\infty = \max_j \alpha_j$, then the error of the sparse grid quadrature is $O(N^{-r}(\log N)^{(d-1)(r-1)})$ [11, 2, 15]. The N here is the number of the sparse grid quadrature points.

Next, we study the accuracy of the dimension-adaptive sparse grid quadrature. The error of the dimension-adaptive sparse grid interpolation has been studied in [14, 13, 12]. Similar as the analysis in [14], we first have following bound on $|Q_L f - Q_I f|$.

Proposition 3.1. (*Priori error bound*) Let $\mathbf{I} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{1}, |\Delta_{\mathbf{i}} f| \geq \epsilon\}$ and $Q_{\mathbf{L}} f - Q_{\mathbf{I}} f$ be the error of the dimension adaptive sparse grid quadrature on set \mathbf{I} relative to $Q_{\mathbf{1}} f$. We further denote $\mathbf{L} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{1}\}$ and the number of all indices in \mathbf{L} is $|\mathbf{L}| = \prod_{k=1}^d (l_k + 1)$. Then we get the bound

$$|Q_{\mathbf{L}} f - Q_{\mathbf{I}} f| \leq |\mathbf{L}| \epsilon. \quad (26)$$

Proof. According to the definition, we have

$$\begin{aligned} |Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| &= \left| \sum_{\mathbf{i} \in \mathbf{L}} \Delta_{\mathbf{i}}f - \sum_{\mathbf{i} \in \mathbf{I}} \Delta_{\mathbf{i}}f \right| = \left| \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} \Delta_{\mathbf{i}}f \right| \\ &\leq \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} |\Delta_{\mathbf{i}}f| \leq \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} \epsilon \leq \sum_{\mathbf{i} \in \mathbf{L}} \epsilon = |\mathbf{L}| \epsilon. \end{aligned} \quad (27)$$

The first inequality follows from the triangle inequality. The second inequality holds because $\mathbf{L} \setminus \mathbf{I} = \{\mathbf{i} \leq \mathbf{l} \mid |\Delta_{\mathbf{i}}f| < \epsilon\}$. The third inequality follows by the fact $\mathbf{I} \subset \mathbf{L}$. \square

From the proof of the proposition (3.1), we do not use any information from the computation process of the $Q_{\mathbf{I}}f$. The error bound can be derived before computing $Q_{\mathbf{I}}f$. However, after we compute $Q_{\mathbf{I}}f$ by the dimension-adaptive sparse grid method, we will know exactly what the downset \mathbf{I} is. This can help us improve the error bound.

Proposition 3.2. (*Posteriori error bound*) Suppose the downset \mathbf{I} is known after we computed the $Q_{\mathbf{I}}f$. The error bound in 3.1 can be improved by

$$|Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| \leq (|\mathbf{L}| - |\mathbf{I}|)\epsilon, \quad (28)$$

where the set $\mathbf{L} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{l}\}$.

Proof. Since $\mathbf{I} = \{\mathbf{i} \mid \mathbf{i} \leq \mathbf{l}, |\Delta_{\mathbf{i}}f| \geq \epsilon\}$ is a subset of \mathbf{L} , we have

$$|Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| = \left| \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} \Delta_{\mathbf{i}}f \right| \leq \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} |\Delta_{\mathbf{i}}f| = (|\mathbf{L}| - |\mathbf{I}|)\epsilon. \quad (29)$$

\square

When we compute a high dimensional integral, we won't set a very small ϵ , e.g. 10^{-15} , in the termination condition since the computational cost is usually unaffordable for most cases. Thus, neither priori error bound nor posteriori error bound is good when we consider a high dimensional problem since $|\mathbf{L}|$ will grow exponentially when the dimension increases while the ϵ can't be chosen as small as possible. In order to get a more accurate error bound, we need to use the smoothness of the integrand f .

Lemma 3.1. If $f \in H^r([-1, 1]^d)$ and we have following estimation for the 1D quadrature rules Q_i

$$|If - Q_i f| \leq \|I - Q_i\| \|f\| \leq \gamma_r 2^{-ri} \|f\| \quad (30)$$

where the constants γ_r can be obtained by known bounds for the respective Peano kernels(ref), then if hierarchical rules are used in building d dimensional quadrature rules, we have $|\Delta_{\mathbf{i}}f| \leq C_r^d 2^{-r|\mathbf{i}|} \|f\|$ where $C_r^d = \gamma_r^d (1 + 2^r)^d$.

Proof. Since we use hierarchical rules, the 1D difference can be written as

$$\begin{aligned} \Delta_i f &= Q_i f - Q_{i-1} f = \sum_{x_{i,j} \in G_i} c_{i,j} f(x_{i,j}) - \sum_{x_{i-1,j} \in G_{i-1}} c_{i-1,j} f(x_{i-1,j}) \\ &= \sum_{x_{i,j} \in G_i} b_{i,j} f(x_{i,j}) \end{aligned} \quad (31)$$

where $b_{i,j} = c_{i,j}$ for quadrature points in the set $G_i \setminus G_{i-1}$ and $b_{i,j} = c_{i,j} - c_{i-1,j}$ otherwise. Then for 2D case, we have

$$\Delta_{i_1} \otimes \Delta_{i_2} f = \sum_{x_{i_1,j_1} \in G_{i_1}} \sum_{x_{i_2,j_2} \in G_{i_2}} b_{i_1,j_1} b_{i_2,j_2} f(x_{i_1,j_1}, x_{i_2,j_2}), \quad (32)$$

Furthermore,

$$\begin{aligned} |\Delta_{i_1} \otimes \Delta_{i_2} f| &\leq \|\Delta_{i_1}\| \sum_{x_{i_2,j_2} \in G_{i_2}} b_{i_2,j_2} f(\cdot, x_{i_2,j_2}) \\ &\leq \|\Delta_{i_1}\| \sup_{0 \leq \alpha_1 \leq r} \sup_{s \in [-1,1]} |\Delta_{i_2} f^{(\alpha_1,0)}(s, \cdot)| \\ &\leq \|\Delta_{i_1}\| \sup_{0 \leq \alpha_1 \leq r} \sup_{s \in [-1,1]} \|\Delta_{i_2}\| \|f^{(\alpha_1,0)}(s, \cdot)\| \\ &\leq \|\Delta_{i_1}\| \|\Delta_{i_2}\| \|f\|. \end{aligned} \quad (33)$$

This can be generated to d dimensional case that is

$$|\Delta_{\mathbf{i}} f| = |\Delta_{i_1} \otimes \Delta_{i_2} \otimes \cdots \otimes \Delta_{i_d} f| \leq \|\Delta_{i_1}\| \|\Delta_{i_2}\| \cdots \|\Delta_{i_d}\| \|f\|. \quad (34)$$

We can derive the upper bound of the norm of the 1d difference operator from

$$\begin{aligned} \|\Delta_{i_k}\| &= \|Q_{i_k} - Q_{i_{k-1}}\| \\ &\leq \|I - Q_{i_k}\| + \|I - Q_{i_{k-1}}\| \\ &\leq r_\gamma 2^{-r i_k} (1 + 2^r). \end{aligned} \quad (35)$$

Combining (34) and (35), we get

$$|\Delta_{\mathbf{i}} f| \leq C_r^d 2^{-r|\mathbf{i}|} \|f\|. \quad (36)$$

□

Theorem 3.1. *Under the conditions of Lemma (3.1), we can further improve our posteriori bound by*

$$|Q_{\mathbf{L}} f - Q_{\mathbf{I}} f| \leq K \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{-r|\mathbf{i}|}. \quad (37)$$

Proof. The proposition (3.1) follows directly from the proposition (3.2) and the Lemma (3.1). $K = C_r^d \|f\|$ is a constant. □

Corollary 3.1. *Suppose \mathbf{m} is one of the indices such that $|\mathbf{m}| \leq |\mathbf{i}|, \forall \mathbf{i} \in \mathbf{L} \setminus \mathbf{I}$, then the posterior bound is*

$$|Q_{\mathbf{L}} f - Q_{\mathbf{I}} f| \leq \frac{\epsilon}{\rho} \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{r(|\mathbf{m}| - |\mathbf{i}|)} \quad (38)$$

where $\rho K 2^{-r|\mathbf{m}|} = \epsilon$ and

$$\rho_{\min} := \frac{\sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{r(|\mathbf{m}| - |\mathbf{i}|)}}{|\mathbf{L}| - |\mathbf{I}|} \leq \rho < 2^r. \quad (39)$$

Proof. We first notice we can rewrite (37) as

$$|Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| \leq K2^{-r|\mathbf{m}|} \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{r(|\mathbf{m}| - |\mathbf{i}|)}. \quad (40)$$

By using $\rho K2^{-r|\mathbf{m}|} = \epsilon$, we get the inequality (38). For the lower bound of ρ , we expect the error bound (38) is not worse than the posterior error bound in the proposition (3.2), otherwise we can use the latter one. Thus, we have

$$\frac{\epsilon}{\rho} \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{r(|\mathbf{m}| - |\mathbf{i}|)} \leq \epsilon(|\mathbf{L}| - |\mathbf{I}|), \quad (41)$$

This leads to the lower bound of ρ . For the upper bound, if we denote the k th negative unit vector as $\mathbf{e}_k = [0, \dots, 1, \dots, 0]$, then according to the definition of \mathbf{m} , there exists an index $\mathbf{m} - \mathbf{e}_k \in \mathbf{I}$, otherwise we should choose $\mathbf{m} - \mathbf{e}_k$ instead of \mathbf{m} in the theorem. According to the Theorem 3.1 and the definition of the downset \mathbf{I} , we have

$$\epsilon < |\Delta_{\mathbf{m} - \mathbf{e}_k} f| \leq K2^{-r(|\mathbf{m}| - 1)}. \quad (42)$$

Thus, using the definition of ρ , we have

$$\rho = \frac{\epsilon}{K2^{-r|\mathbf{m}|}} \leq \frac{K2^{-r(|\mathbf{m}| - 1)}}{K2^{-r|\mathbf{m}|}} = 2^r. \quad (43)$$

□

By using the error estimation in the Theorem 3.1 and the error bound for d -dimensional product rule, we can obtain a bound on $|If - Q_{\mathbf{I}}f|$ by using trapezoidal rule, that is

$$|If - Q_{\mathbf{I}}f| \leq |If - Q_{\mathbf{L}}f| + |Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| \leq c_d 2^{-lr} + K \sum_{\mathbf{i} \in \mathbf{L} \setminus \mathbf{I}} 2^{-r|\mathbf{i}|} \quad (44)$$

In the above bound, isotropic grid \mathcal{G}_1 , $l_i = l$ is used in the comparison. In [13], the authors get an optimised priori error bound for $|If - Q_{\mathbf{I}}f|$ by balancing error bounds for the term $|If - Q_{\mathbf{L}}f|$ and $|Q_{\mathbf{L}}f - Q_{\mathbf{I}}f|$. The ϵ need to be chosen very small in order to achieve the optimized bound for high dimensional problems. In this paper, we are more interested in the case when

$$|If - Q_{\mathbf{L}}f| \ll |Q_{\mathbf{L}}f - Q_{\mathbf{I}}f| \quad (45)$$

which means the approximation of the integral need to be accurate to some extent on the corresponding full grid, otherwise we can not expect the dimensional adaptive sparse grid method which uses a subset of quadrature points on the full grid provides a good approximation. Larger ϵ is allowed in this situation.

4 The 'optimise then discretise' method

We will show the framework of the 'optimise then discretise' method and test a 2D example to illustrate its performance. For simplicity, we first use Newton method as our optimisation algorithm. It is shown in the Algorithm 2. The Algorithm 3, 4, 5, 6 and 7 are discretised versions of the Newton method in

Algorithm 2 from simple to complex. In the Algorithm 3 and 4, we use non-adaptive quadrature Q to compute the integrals. In the Algorithm 6 and 7, we use the dimension adaptive quadrature \mathcal{Q} to compute the integrals. The notation D_i denotes the i th discretised derivative and thus D_{ij}^2 is the ij th second order discretised derivative. D denotes the discretised gradient and D^2 denotes the discretised Hessian.

In Algorithm 3, we use different non-adaptive surrogate at each iteration. The non-adaptive quadrature operator Q and the discretised derivative operators D_i , D_{ij} are commutative, i.e.

$$D_i Q_p = Q_p D_i \text{ and } D_{ij}^2 Q_p = Q_p D_{ij}^2.$$

This is because both two operators are fixed finite summations. In Algorithm 4, we further allow different choices of the non-adaptive quadrature for the objective and different component of gradient and Hessian.

The Algorithm 5 looks almost the same as the Algorithm 3 except the quadrature method is dimension adaptive. However, they have essential differences. The dimension adaptive operator \mathcal{Q} and the discretised derivative operators D_i , D_{ij} are not commutative. This observation leads to a new Algorithm 6. The reason why the dimension adaptive operator and the discretised derivative operators are not commutative is because the downsets used in the computation are not equal, i.e.

$$\begin{aligned} I_{(D_i f, \epsilon_p, \mathbf{1}_p)} &\neq I_{(f, \epsilon_p, \mathbf{1}_p)} \\ I_{(D_{ij}^2 f, \epsilon_p, \mathbf{1}_p)} &\neq I_{(f, \epsilon_p, \mathbf{1}_p)}. \end{aligned}$$

One can also generalise the Algorithm 6 to the Algorithm 7 by allowing the usage of different parameters in dimension adaptive quadrature for the objective and different component of gradient and Hessian. Though Algorithm 7 will be more flexible than Algorithm 6, we mostly use Algorithm 6 in practice because it is usually hard to get information used for choose different parameters and the algorithm 7 is too complex.

Algorithm 2 OPTIMISE

- 1: Take an initial $u_0 \in \mathbb{R}^n$ and $p := 0$
- 2: Compute $G_0 = \nabla F(u_0)$
- 3: **while** $\|G_p\| > \epsilon$ **do**
- 4: Compute the Hessian $H_p = \nabla^2 F(u_p)$
- 5: Update

$$u_{p+1} = u_p - H_p^{-1} G_p$$

- 6: Set $p := p + 1$
 - 7: Compute $G_p = \nabla F(u_p)$
 - 8: **end while**
 - 9: Output u_p and $F(u_p)$
-

For more complicated quasi Newton methods, we take BFGS method with the exact line search as an example. The optimise algorithm and its adaptive discretised version are shown in the Algorithm 8 and Algorithm 9 which are

Algorithm 3 DISCRETISED VERSION

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$ and $p := 0$
- 2: Compute the approximation of the gradient $\bar{G}_0 = DQ_0(f(\bar{u}_0, \cdot))$
- 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
- 4: Compute the approximation of the Hessian $\bar{H}_p = D^2Q_p(f(\bar{u}_p, \cdot))$
- 5: Update

$$\bar{u}_{p+1} = \bar{u}_p - \bar{H}_p^{-1} \bar{G}_p \quad (46)$$

- 6: Set $p := p + 1$
 - 7: Compute the approximation of the gradient $\bar{G}_p = DQ_p(f(\bar{u}_p, \cdot))$
 - 8: **end while**
 - 9: Output \bar{u}_p and $\bar{F}_p := Q_p(f(\bar{u}_p, \cdot))$
-

Algorithm 4 GENERAL DISCRETISED VERSION

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$ and $p := 0$
- 2: Compute the approximation of the gradient $\bar{G}_0 = [Q_{0,i}^G(D_i f(\bar{u}_0, \cdot))]_{n \times 1}$
- 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
- 4: Compute the approximation of the Hessian $\bar{H}_p = [Q_{p,i,j}^H(D_{ij}^2 f(\bar{u}_p, \cdot))]_{n \times n}$

- 5: Update

$$\bar{u}_{p+1} = \bar{u}_p - \bar{H}_p^{-1} \bar{G}_p \quad (47)$$

- 6: Set $p := p + 1$
 - 7: Compute the approximation of the gradient $\bar{G}_p = [Q_{p,i}^G(D_i f(\bar{u}_p, \cdot))]_{n \times 1}$
 - 8: **end while**
 - 9: Output \bar{u}_p and $\bar{F}_p := Q_p^O(f(\bar{u}_p, \cdot))$
-

Algorithm 5 DISCRETISED VERSION(ADAPTIVE)

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$ and $p := 0$
- 2: Compute the approximation of the gradient $\bar{G}_0 = DQ_{\epsilon_0, \mathbf{l}_0}(f(\bar{u}_0, \cdot))$
- 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
- 4: Compute the approximation of the Hessian $\bar{H}_p = D^2Q_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$
- 5: Update

$$\bar{u}_{p+1} = \bar{u}_p - \bar{H}_p^{-1} \bar{G}_p \quad (48)$$

- 6: Set $p := p + 1$
 - 7: Compute the approximation of the gradient $\bar{G}_p = DQ_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$
 - 8: **end while**
 - 9: Output \bar{u}_p and $\bar{F}_p := Q_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$
-

Algorithm 6 MODIFIED DISCRETISED VERSION(ADAPTIVE)

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$ and $p := 0$
 - 2: Compute the approximation of the gradient $\bar{G}_0 = [\mathcal{Q}_{\epsilon_0, \mathbf{l}_0}(D_i f(\bar{u}_0, \cdot))]_{n \times 1}$
 - 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
 - 4: Compute the approximation of the Hessian $\bar{H}_p = [\mathcal{Q}_{\epsilon_p, \mathbf{l}_p}(D_{ij}^2 f(\bar{u}_p, \cdot))]_{n \times n}$
 - 5: Update
$$\bar{u}_{p+1} = \bar{u}_p - \bar{H}_p^{-1} \bar{G}_p \quad (49)$$
 - 6: Set $p := p + 1$
 - 7: Compute the approximation of the gradient $\bar{G}_p = [\mathcal{Q}_{\epsilon_p, \mathbf{l}_p}(D_i f(\bar{u}_p, \cdot))]_{n \times 1}$
 - 8: **end while**
 - 9: Output \bar{u}_p and $\bar{F}_p := \mathcal{Q}_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$
-

Algorithm 7 GENERAL DISCRETISED VERSION(ADAPTIVE)

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$ and $p := 0$
 - 2: Compute the approximation of the gradient
$$\bar{G}_0 = [\mathcal{Q}_{\epsilon_0, i, \mathbf{l}_{0, i}}^G(D_i f(\bar{u}_0, \cdot))]_{n \times 1}$$
 - 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
 - 4: Compute the approximation of the Hessian
$$\bar{H}_p = [\mathcal{Q}_{\epsilon_p, i, j, \mathbf{l}_{p, i, j}}^H(D_{ij}^2 f(\bar{u}_p, \cdot))]_{n \times n}$$
 - 5: Update
$$\bar{u}_{p+1} = \bar{u}_p - \bar{H}_p^{-1} \bar{G}_p \quad (50)$$
 - 6: Set $p := p + 1$
 - 7: Compute the approximation of the gradient
$$\bar{G}_p = [\mathcal{Q}_{\epsilon_p, i, \mathbf{l}_{p, i}}^G(D_i f(\bar{u}_p, \cdot))]_{n \times 1}$$
 - 8: **end while**
 - 9: Output \bar{u}_p and $\bar{F}_p := \mathcal{Q}_{\epsilon_p, \mathbf{l}_p}^O(f(\bar{u}_p, \cdot))$
-

extensions of the Algorithm 2 and the Algorithm 6, respectively. In practice, the exact line search (51) is replaced with inexact line search for efficiency. The commonly used inexact line search is strong Wolfe's rule. The sequence u_p generated by BFGS with Wolfe's rule is proved to converge to the exact minimizer u^* superlinearly [17]. It should be noted that we need to compute the objective F in the line search methods(include the strong Wolfe's rule used in practice)in each iteration while this is not required if we use Newton method.

Algorithm 8 BFGS OPTIMISE

- 1: Take an initial $u_0 \in \mathbb{R}^n$, an initial positive definite matrix H_0 and $p := 0$
- 2: **while** $\|G(u_p)\| > \epsilon$ **do**
- 3: Compute the search direction $v_p = -H_p G(u_p)$
- 4: Find the step length α_p by exact line search

$$\min_{\alpha_p} F(u_p + \alpha_p v_p). \quad (51)$$

The underlying A_p^{-1} here is $\alpha_p H_p$.

- 5: Update

$$u_{p+1} = u_p + \alpha_p v_p$$

- 6: Define $s_p := u_{p+1} - u_p$ and $y_p := G(u_{p+1}) - G(u_p)$
- 7: Update

$$H_{p+1} = \left(I - \frac{s_p y_p^T}{s_p^T y_p} \right) H_p \left(I - \frac{y_p s_p^T}{s_p^T y_p} \right) + \frac{s_p s_p^T}{s_p^T y_p}$$

- 8: $p := p+1$
 - 9: **end while**
 - 10: Output u_p and $F(u_p)$
-

In order to illustrate our method, we provide the following 2D example. We will look into this example and it will also be used to explain the idea of next two sections.

Example 4.1. We consider the following minimization problem

$$\min_{u \in \mathbb{R}} F(u) \quad (52)$$

where $F(u) = \mathbb{E} [u^2 + (W_1^2 + 10W_2^2)u]$. W_1 and W_2 are i.i.d. random variables. The integrand satisfies assumption 2.1. Moreover, the objective function is strictly convex in this example, so we conclude that there is a unique minimizer of this problem. By using the linearity of the expectation, the minimizer of the problem is

$$u^* = -\frac{\mathbb{E}[W_1^2] + 10\mathbb{E}[W_2^2]}{2}. \quad (53)$$

In particular, here we further assume

$$W_k \sim \text{Beta}(\alpha, \beta), \quad k = 1, 2. \quad (54)$$

with $\alpha = 5, \beta = 5$. The exact minimizer is then $u^* = -1.5$ and the minimum is -2.25 .

Algorithm 9 BFGS DISCRETISE

- 1: Take an initial $\bar{u}_0 \in \mathbb{R}^n$, an initial positive definite matrix \bar{H}_0 and $p := 0$
- 2: Compute $\bar{G}_0 = [\mathcal{Q}_{\epsilon_0, \mathbf{l}_0}(D_i f(\bar{u}_0, \cdot))]_{n \times 1}$
- 3: **while** $\|\bar{G}_p\| > \epsilon$ **do**
- 4: Compute the search direction $\bar{v}_p = -\bar{H}_p \bar{G}_p$
- 5: Find the step length $\bar{\alpha}_p$ by exact line search

$$\min_{\bar{\alpha}_p} \bar{F}_p(\bar{u}_p + \bar{\alpha}_p \bar{v}_p)$$

- 6: where $\bar{F}_p := \mathcal{Q}_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$ and the corresponding \bar{A}_p^{-1} is $\bar{\alpha}_p \bar{H}_p$
- 6: Update

$$\bar{u}_{p+1} = \bar{u}_p + \bar{\alpha}_p \bar{v}_p$$

- 7: Compute $\bar{G}_{p+1} = [\mathcal{Q}_{\epsilon_{p+1}, \mathbf{l}_{p+1}}(D_i f(\bar{u}_{p+1}, \cdot))]_{n \times 1}$
- 8: Define $\bar{s}_p := \bar{u}_{p+1} - \bar{u}_p$ and $\bar{y}_p := \bar{G}_{p+1} - \bar{G}_p$
- 9: Update

$$\bar{H}_{p+1} = \left(I - \frac{\bar{s}_p \bar{y}_p^T}{\bar{s}_p^T \bar{y}_p} \right) \bar{H}_p \left(I - \frac{\bar{y}_p \bar{s}_p^T}{\bar{s}_p^T \bar{y}_p} \right) + \frac{\bar{s}_p \bar{s}_p^T}{\bar{s}_p^T \bar{y}_p}$$

- 10: Set $p := p+1$
 - 11: **end while**
 - 12: Output \bar{u}_p and $\bar{F}_p := \mathcal{Q}_{\epsilon_p, \mathbf{l}_p}(f(\bar{u}_p, \cdot))$
-

In Figure 1, we apply the Algorithm 9 to solve the problem. We use forward difference to approximate the derivatives of the integrand. We compare the performance of the dimension-adaptive sparse grid quadrature and the sparse grid quadrature. For the OTDM based on the sparse grid quadrature, we fix the level l for each run which results in the same choices of the quadrature rule Q_p and $Q_{p,1}$ for any p . For the OTDM based on the dimension-adaptive sparse grid quadrature, we also fix the ϵ and \mathbf{l} in the while condition 21. However, the choices of the quadrature rule are no longer the same for all Q_p and $Q_{p,1}$. This is because the underlying downsets are not necessary to be the same for all Q_p and $Q_{p,1}$. We see in Figure 1 that the convergence rates are improved for both Clenshaw–Curtis and Gauss–Patterson when we apply the dimension-adaptive method. Especially, the average number of the grid points used in each iteration is substantially reduced for the same accuracy in the solution and the objective when Gauss–Patterson is used as 1D rule. For Clenshaw–Curtis, we can also see this pattern, moreover, higher accuracy are obtained for both computed minimizer and minimum.

The exact expression of the gradient in the previous 2D Example 4.1 is

$$\begin{aligned} G(u) &= \nabla \mathbb{E}[u^2 + (W_1^2 + 10W_2^2)u] \\ &= \int_0^1 \int_0^1 [2u + (w_1 + 10w_2)] p(w_1, \alpha, \beta) p(w_2, \alpha, \beta) dw_1 dw_2 \end{aligned} \quad (55)$$

where p is the probability density function. In Figure 4, we again solved problem with the BFGS method. We use 100 points and 1000 points Monte Carlo method

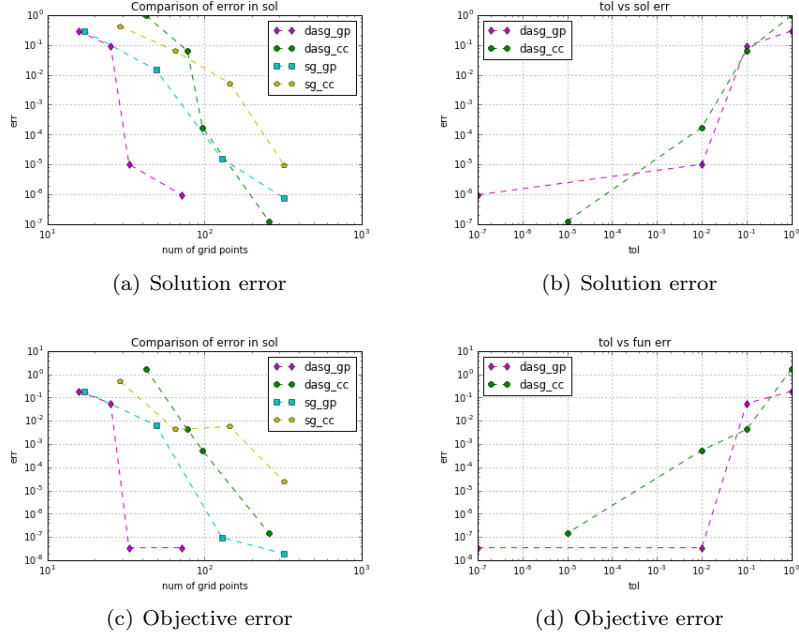


Figure 1: Computational results for the 2D problem: (a) errors of computed minimizer vs. number of grid points used in each iteration (average for the dimension-adaptive sparse grid method). (b) errors of computed minimizer vs. ϵ in the termination condition of the dimension-adaptive algorithm. (c) errors of computed minimum vs. number of grid points used in each iteration. (d) errors of computed minimum vs. ϵ in the termination condition of the dimension-adaptive algorithm. We compare the dimension-adaptive sparse grid based on Gauss–Patterson (dasg_gp) and Clenshaw–Curtis (dasg_cc) with sparse grid based on Gauss–Patterson (sg_gp) and Clenshaw–Curtis (sg_cc).

to approximate the integral $G(u)$ respectively. The sparse grid, the dimensional-adaptive sparse grid based on Gauss–Patterson 1D quadrature and the Monte Carlo are used in approximating the objective function. We intentionally choose the same random points for Monte Carlo in objective approximation with those in gradient approximation when number of points equals to 100 in (a), (b) and 1000 in (c), (d). Thus, according to the propositions, the Monte Carlo surrogate methods are actually used. We can see from the Figure 4, both convergence performances of sparse grid and dimensional adaptive sparse grid are better than the Monte Carlo method (include those points which is actually surrogate method). The amount of work will be substantially reduced in the objective function if we apply the dimension-adaptive sparse grid in computing the objective. In addition, the dimension-adaptive sparse grid method performs better than sparse grid method in computing both solution and objective.

5 Convergence analysis

We now study the sequence generated by the algorithm introduced in the previous section. Because of the presence of the errors (Both truncated errors and rounding errors), the actual sequences $\{\bar{u}_p\}$ generated by the Newton-type

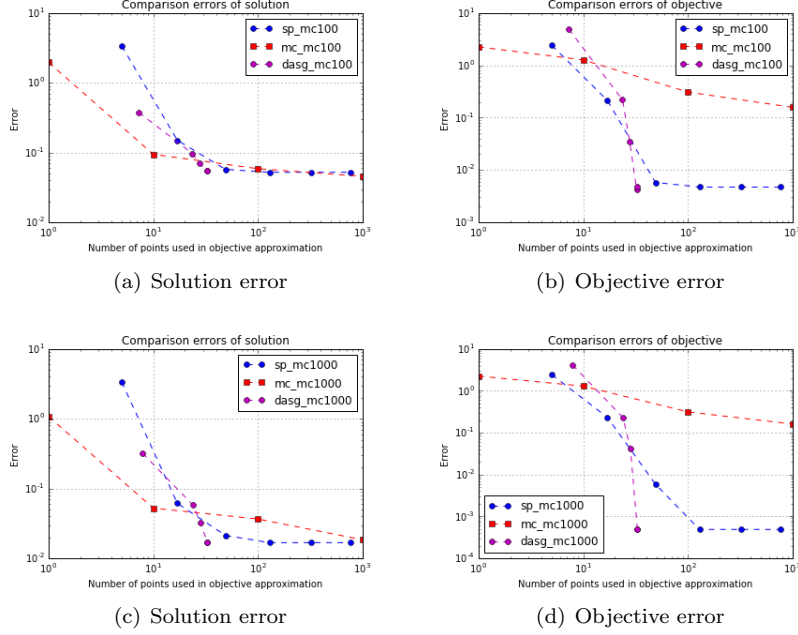


Figure 2: Computational results for the toy problem. Here we use the exact expression of the gradient. Monte Carlo method is used to approximate the gradient and three different quadrature methods are used to approximate the objective function. (a) number of grid points vs. errors of computed minimizers(average of 10 runs of each method with gradient approximated by 100 points Monte Carlo method. (b)number of grid points vs. errors of computed minimum. (c)number of grid points vs. errors of computed minimizers(1000 points Monte Carlo). (d)number of grid points vs. errors of computed minimum.

method is

$$\bar{u}_{p+1} = (I + E_0(\bar{u}_p)) (\bar{u}_p + \bar{\theta}(\bar{u}_p)), \quad p = 0, 1, \dots, \quad (56)$$

In the above formula, $E_0(\bar{u}_p)$ is the relative error of the addition of \bar{u}_p and $\bar{\theta}(\bar{u}_p)$. The norm of this error is of the order of the machine epsilon and thus very small, so this error is neglected in our analysis. $\bar{\theta}(\bar{u}_p)$ is the exact solution of the following linear system

$$\bar{A}_p z = -\bar{G}_p \quad (57)$$

where

$$\begin{aligned} \bar{A}_p &= \hat{A}_p + E_1(\bar{u}_p) \\ \bar{G}_p &= G(\bar{u}_p) + E_2(\bar{u}_p). \end{aligned} \quad (58)$$

\hat{A}_p is an approximation of the Hessian. Since \hat{A}_p is computed by using the information of previously computed \bar{u}_t and \hat{A}_t , $t < p$, we use the different notation \hat{A}_p rather than A_p here. $E_1(\bar{u}_p)$ and $E_2(\bar{u}_p)$ are errors occur when we approximate \hat{A}_p and $G(\bar{u}_p)$.

In [25], the author shows the crucial condition for the convergence of such perturbed Newton-type method is

$$\eta_p = \|\bar{A}_p^{-1} G'(\bar{u}_p) - I\| + \frac{\|\bar{A}_p^{-1} (\bar{G}_p - G_p)\|}{\|(G'(\bar{u}_p))^{-1} G_p\|} < 1 \quad (59)$$

The following theorem [25] provides more details on the convergence result

Theorem 5.1. *If $\|E_0\| = 0$ and $\eta_p \leq \eta < 1$ for all $p = 0, 1, \dots$, and if \bar{u}_0 satisfies*

$$\|\bar{u}_0 - u^*\| < \frac{2(1-\eta)}{(3-\eta)\mu}, \quad (60)$$

then the perturbed Newton-type method (56) – (58) produces a sequence $\{\bar{u}_p\}$ which converges to u^ .*

Because of the presence of the term $\|(G'(\bar{u}_p))^{-1}G_p\|$ in the denominator, we can expect that the inequality (59) will finally be violated and the convergence theory will fail. Therefore, there exists some neighborhood of u^* such that the sequence generated by the Newton-type method converges outside this neighborhood while the behavior of the sequence generated by the successive iterations is not predictable if they are inside the neighborhood. This also suggests further iterations will not improve the accuracy and we can stop the algorithm once the \bar{u}_p reaches the neighborhood.

In order to find out when to stop further iterations, we look into the two terms in (59). When $G'(\bar{u}_p)$ can be well approximated and (57) can be solved with relative high accuracy, the first term can be kept small. This means the errors in the Jacobian approximation are tolerable if

$$\tau_p = \|\bar{A}_p^{-1}G'(\bar{u}_p) - I\| \ll 1. \quad (61)$$

Corollary 5.1. *[25] If (61) holds, then the convergence of the perturbed Newton-type method might breakdown when*

$$\|\bar{G}_p - G_p\| > \frac{\|\bar{G}_p\|}{1 + \kappa(\bar{A}_p)}. \quad (62)$$

where $\kappa(\bar{A}_p)$ is the condition number of \bar{A}_p .

According to the Corollary 5.1, the accuracy of computed minimizer depends on how accurate the gradient can be approximated. For the same problem, if the gradients can be approximated in higher accuracy, which means we can get smaller $\|\bar{G}_p - G_p\|$, then it is likely that more effective iterations can be carried on in the Newton-type method according to (62). Thus, we can obtain a better solution. The problem is it will be very expensive to get a moderately accurate approximation to the gradient if we are dealing with high dimensional problems. Thus, there is no need to do iteration (8) many times for this kind of problems and (62) actually provide us with a simple stopping criterion. In the stopping criterion (62), the condition number $\kappa(\bar{A}_p)$ can be computed during the iteration with little cost. However, it will be difficult to obtain the exact value $\|\bar{G}_p - G_p\|$ and $\|G_p\|$. Following theorem gives an estimation of the ratio of $\|\bar{G}_p - G_p\|$ to $\|G_p\|$ and provide more detailed stopping criterion for our method.

Theorem 5.2. *Consider solving the stochastic optimisation problem (2), we assume here the integrand satisfies*

- (i) $f(\cdot, W) \in C^2(U)$ for all $W \in \Omega$.
- (ii) $f(u, \cdot) \in H^r(\Omega)$ ($r \geq 1$) for all $u \in U$.

We apply the dimension-adaptive sparse grid(ϵ) to approximate the integrals and forward difference method to approximate the derivatives in the computation. We further assume all the components of the gradient at p th iteration are computed by the same downset as that in computing the objective function at p th iteration. Then, we have

$$\|\bar{G}_p - G_p\| \leq \|\mathcal{E}_p^1\| + \|\mathcal{E}_p^2\| \quad (63)$$

where the q th element of \mathcal{E}_p^1 and \mathcal{E}_p^1 are

$$\begin{aligned} \mathcal{E}_{p,q}^1 &= K_{p,q}^1 h \\ \mathcal{E}_{p,q}^1 &= K_{p,q}^2 2^{-l_p r} + K_{p,q}^3 \sum_{\mathbf{i} \in \mathbf{L}_p \setminus \mathbf{I}_p} 2^{-|\mathbf{i}|r} \end{aligned} \quad (64)$$

where $K_{p,q}^i$, $i = 1, 2, 3$ are constants. Therefore, our method based on dimension-adaptive sparse grid method might breakdown when

$$\|\mathcal{E}_p^1\| + \|\mathcal{E}_p^2\| > \frac{\|\bar{G}_p\|}{1 + \kappa(\bar{A}_p)}. \quad (65)$$

Proof. We use forward difference quotient

$$\tilde{G}_{p,q} = \frac{F(\bar{u}_p + h\mathbf{e}_q) - F(\bar{u}_p)}{h} \quad (66)$$

to approximate the q th component $G_{p,q}$ of G_p , where h is the difference increment. We further denote the q th component of \bar{G}_p as $\bar{G}_{p,q}$ which can be expressed as

$$\bar{G}_{p,q} = \frac{\bar{F}(\bar{u}_p + h\mathbf{e}_q) - \bar{F}(\bar{u}_p)}{h} \quad (67)$$

where $\bar{F}(\bar{u}_p) = Q_{\mathbf{I}_p}(f(\bar{u}_p, \cdot))$.

By using triangle inequality, we have

$$\|\bar{G}_p - G_p\| \leq \|\bar{G}_p - \tilde{G}_p + \tilde{G}_p - G_p\| \leq \|\bar{G}_p - \tilde{G}_p\| + \|\tilde{G}_p - G_p\|. \quad (68)$$

Since $f(\cdot, W) \in C^2(U)$ for any $W \in \Omega$, we can get $F \in C^2(U)$. Thus, for the q th component of $\tilde{G}_p - G_p$, we have

$$|\tilde{G}_{p,q} - G_{p,q}| \leq K_{p,q}^1 h. \quad (69)$$

For the term $\|\bar{G}_p - \tilde{G}_p\|$, we have

$$\begin{aligned} & \bar{G}_{p,q} - \tilde{G}_{p,q} \\ &= \frac{1}{h}(\bar{F}(\bar{u}_p + h\mathbf{e}_q) - \bar{F}(\bar{u}_p)) - \frac{1}{h}(F(\bar{u}_p + h\mathbf{e}_q) - F(\bar{u}_p)) \\ &= \frac{1}{h}(Q_{\mathbf{I}_p}(f(\bar{u}_p + h\bar{e}_q, \cdot)) - Q_{\mathbf{I}_p}(f(\bar{u}_p, \cdot))) - \frac{1}{h}(I(f(\bar{u}_p + h\bar{e}_q, \cdot)) - I(f(\bar{u}_p, \cdot))) \\ &= Q_{\mathbf{I}_p} \left(\frac{1}{h}(f(\bar{u}_p + h\bar{e}_q, \cdot) - f(\bar{u}_p, \cdot)) \right) - I \left(\frac{1}{h}(f(\bar{u}_p + h\bar{e}_q, \cdot) - f(\bar{u}_p, \cdot)) \right) \\ &= (Q_{\mathbf{I}_p} - I) \left(\frac{1}{h}(f(\bar{u}_p + h\bar{e}_q, \cdot) - f(\bar{u}_p, \cdot)) \right). \end{aligned} \quad (70)$$

We used the assumption of the same downset I_p in the second equality. The third equality is due to the linearity of the operator $Q_{\mathbf{I}_p}$ and I . Since $f(u, \cdot) \in H^r(\Omega)$ for any $u \in U$, we know that the function

$$\frac{1}{h}(f(\bar{u}_p + h\bar{e}_q, \cdot) - f(\bar{u}_p, \cdot)) \in H^r(\Omega). \quad (71)$$

Thus, we get following upper bound

$$|\bar{G}_{p,q} - \tilde{G}_{p,q}| \leq K_{p,q}^2 2^{-l_p r} + K_{p,q}^3 \sum_{\mathbf{i} \in \mathbf{I}_p \setminus \mathbf{I}_p} 2^{-|\mathbf{i}|r} \quad (72)$$

Combining inequalities (68), (72) and (69), we get (63).

By using Corollary 5.1, we get the breakdown condition (65). \square

Remark 5.1. *Theorem (5.2) actually provides us with rough stopping criterion. It can be checked at each iteration in Newton-type method if one can get reasonable estimations of the constants $K_{p,q}^1$, $K_{p,q}^2$, $K_{p,q}^3$. The h , l_p , \mathbf{L}_p and \mathbf{I}_p can be obtained during the computation. The second terms of $\mathcal{E}_{p,q}^2$ can also be replaced by the bound in corollary (3.1). The advantage of doing this is that instead of estimating the constant $K_{p,q}^3$, we can get a reasonable stop by tuning ρ .*

6 Stopping Criterion

In this section, we discuss the stopping criterion for our method. It is important to know when we should stop the Newton-type method for fixed tolerance ϵ in the termination condition of the dimension-adaptive sparse grid method. A good stopping criterion can save a lot of computational cost. This is because if the problem is high dimensional, each iteration will be very expensive to compute even if we apply the dimension-adaptive sparse grid method to reduce the cost in computing the related integrals. Another reason is it is possible that further iterations can not improve the accuracy of the solution. It might happen that the computed solution becomes even worse after we increase more iterations. Also, computation of the termination condition of some existing optimization solvers can involve a great number of evaluations of objective functions and gradients, such as the Wolfe's rule used in BFGS method in `scipy.optimize` package. Sometimes, computing the termination condition is even expensive than computing the solution itself. The study of the stopping criterion can also shed light on how to choose the tolerance ϵ in the termination condition of the dimension-adaptive sparse grid method for a specific problem with given requirement on the accuracy of the solution.

Though Theorem (5.2) gives us some hints to derive a stopping criterion, accurate estimations of the coefficients are required. However, this is usually difficult especially for high dimensional problems. In addition, the estimation method can vary for different problems, thus it will be complex to obtain a general stopping criterion from the Theorem.

Here we provide another way to find the stopping criterion. It works better and much easier to implement in practice. Suppose \bar{u}_p is the approximated minimizer generated by some Newton-type methods after p th iteration. When \bar{u}_p is close enough to the exact minimizer u^* , we have following Taylor expansion

$$F(\bar{u}_p) = F(u^*) + \nabla F(u^*)(\bar{u}_p - u^*) + (\bar{u}_p - u^*)^T \nabla^2 F(u^*)(\bar{u}_p - u^*) + o(\|\bar{u}_p - u^*\|^2). \quad (73)$$

Since u^* is the minimizer of F , we have $\nabla F(u^*) = 0$. F is convex, therefore $\nabla^2 F(u^*)$ is positive semidefinite. Thus, we have $\|\bar{u}_p - u^*\|^2$ increases(decreases) with i as $F(\bar{u}_p)$ increases(decreases). However, the exact value of $F(\bar{u}_p)$ is usually not easy to get. Therefore, in our approach, instead of using the exact function value at \bar{u}_p , we use the value of some high accuracy approximations of the function. If we denote the high accuracy approximation by dimension-adaptive sparse grid quadrature with ϵ in its termination condition at u as $F_\epsilon(u) = Q_{\mathbf{I}}f(u, \cdot)$, then we can decide when to stop the Newton-type method by studying the trend of $F_\epsilon(\bar{u}_p)$.

The advantage of this stopping criterion is the computation does not involve estimation of the error $\|G - \bar{G}\|$. We only need to compute $F_\epsilon(\bar{u}_p)$ for some smaller ϵ s with the same algorithm which used to compute the approximation of the objective function. Also, the additional computational cost for computing $F_\epsilon(\bar{u}_p)$ is affordable in most cases for even high dimensional problems. This is because we use Newton-type method as our optimisation solver, so the number of iterations will not be too large. Also, the computational cost of getting such stopping criterion is much lower than that of computing the gradient in each iteration when the dimension of U is high.

In fig 3 and fig 4, we solve the toy problem by using surrogate method with $\epsilon = 1$ and $\epsilon = 0.1$ respectively. For both two figures, the subfigures in the first row show the relation between error $|u^* - \bar{u}_p|$ versus the number of iterations. We can see from these figures, we should stop the algorithm after first iteration for all three cases. If we further increase the iteration, the errors will grows. The reason for this is the gradients are approximated with relatively low accuracy. From the convergence theory, we know the convergence of the Newton-type method might break down in this case. The subfigures in the second row presents function values on the surrogate versus the number of iterations. As we expected, $F_\epsilon(\bar{u}_p)$ is decreasing. The subfigures in the third row show the function values of the surrogate functions with $\epsilon = 0.001$ on \bar{u}_p . Compare the subfigures in the first row with the corresponding subfigures in the third row, we can see that the trends of the functions are the same for all three quadrature rules. Thus, we can predict when to stop the Newton-type method by studying the trends of the functions in the third row respectively. Our method successfully predicts the stopping times(after first iteration) for all three quadrature rules in this example. The subfigures in the fourth row, we fit the data points $(|u^* - \bar{u}_p|, F_{\epsilon/4}(\bar{u}_p))$ to a quadratic function. We find that the quadratic curves almost go through all the data points which suggests our error model is reasonable.

7 High dimensional examples

7.1 50D additive example

Consider following minimization problem

Example 7.1.

$$\min_{u \in U} \mathbb{E} \left[\sum_{i=1}^d \exp(-u_i W_i^2) \right] \quad (74)$$

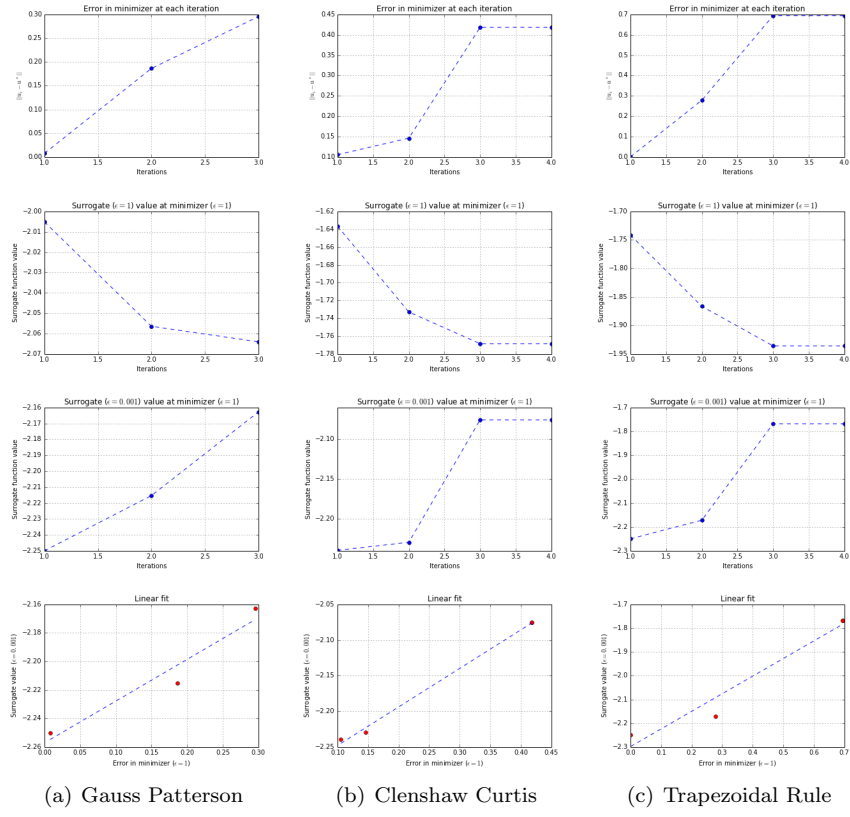


Figure 3: Solve the toy problem with $\epsilon = 1$.

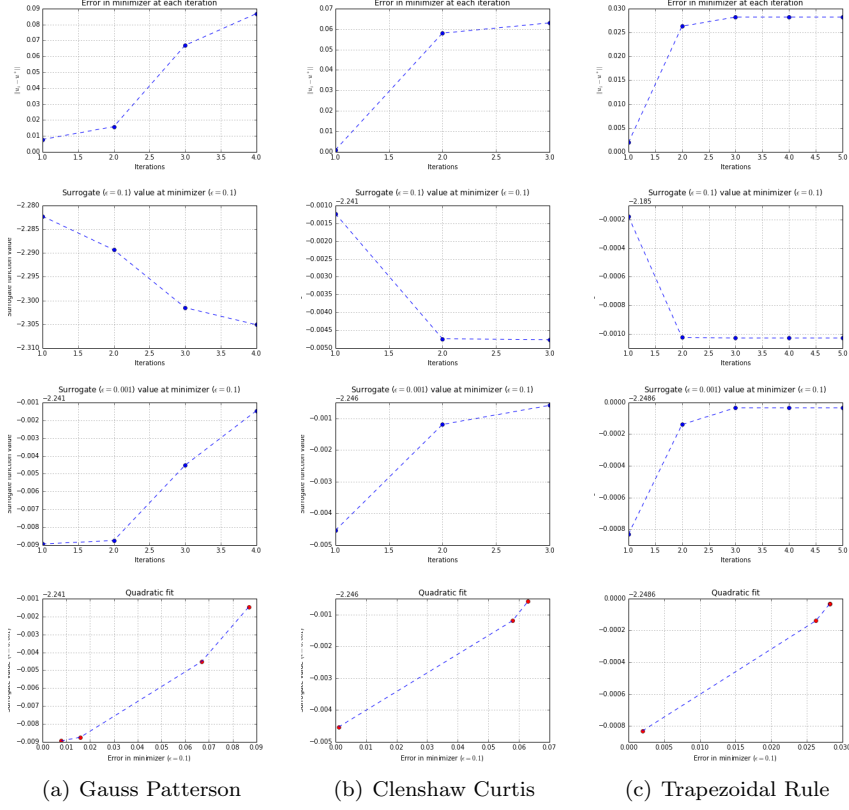


Figure 4: Solve the toy problem with $\epsilon = 0.1$.

where W_i are i.i.d random variables which subject to uniform distribution $U(0, 1)$ and the set $U = [0, 1]^d$. Thus the integral form of the objective function can be written as

$$F(u) = \int_{[0,1]^d} \sum_{i=1}^d e^{-u_i w_i^2} dw. \quad (75)$$

The gradient $G(u)$ is

$$\begin{aligned} G(u) &= \nabla F(u) \\ &= - \left[\int_{[0,1]^d} w_1^2 e^{-u_1 w_1^2} dw_1, \dots, \int_{[0,1]^d} w_d^2 e^{-u_d w_d^2} dw_d \right], \end{aligned} \quad (76)$$

so we have $G(u) \leq \mathbf{0}$ for any $u \in [0, 1]^d$ and thus the exact minimizer is $u^* = (1, \dots, 1)$ for this problem.

The reference objective function value can be computed by

$$F(u) = \int_{[0,1]^d} \sum_{i=1}^d e^{-u_i w_i^2} dW = \sum_{i=1}^d \int_{[0,1]} e^{-u_i w_i^2} dw_i. \quad (77)$$

At the minimizer, we have the exact objective

$$F(u^*) = d \int_{[0,1]} e^{-w_i^2} dw_i = d(\mathcal{F}(1) - \mathcal{F}(0)) \quad (78)$$

where \mathcal{F} is the cumulative distribution function.

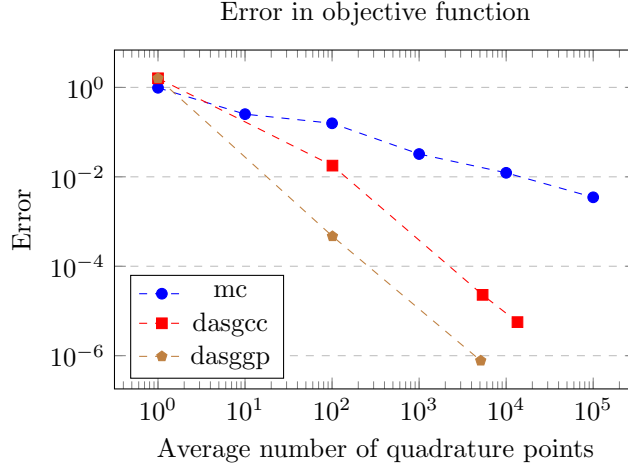


Figure 5: Compute the additive example with $d = 50$.

Here, we apply the 'optimize then discretise' approach to solve the problem. We use 'L-BFGS-B' method in `scipy.optimize` package as our solver. We apply the dimension-adaptive sparse grid method to computing the objective function while use the Monte-Carlo method to approximate the high dimensional integrals involved in computing the gradient.

We still have the approximated gradient $\bar{G}(u) \leq 0$ for any $u \in [0, 1]^d$. This is because the integrand of each entry of the gradient is non-positive and only positive weights are used in the Monte Carlo method. Thus, no matter how many samples used in the Monte Carlo method, we will always get a descent direction at each step during optimization process. It is noteworthy here that both low level sparse grid method and dimensional-adaptive sparse grid method with large ϵ may change the sign of integral approximated and therefore lead to wrong search direction.

In the example, we only use the Monte Carlo method with 10 samples to compute the gradient components. In order to increase the accuracy in minimum, when we approximate the objective function, we increase samples in the Monte Carlo method and decrease ϵ in the termination condition of the dimension-adaptive sparse grid. The result shows all of three methods achieve the exact minimizer as we expected. The errors in minimum of two dimension-adaptive approaches drop much faster than the Monte Carlo method.

7.2 Application to stochastic control

In this section, we illustrate our dimension-adaptive sparse grid method with an instance of a discrete time open-loop stochastic control problem. The general

form of such control problem can be found in Bertsekas [1]. The control problem is described by following discrete time dynamic system

$$x_{i+1} = \psi_i(x_i, u_i, w_i), \quad i = 0, \dots, d-1 \quad (79)$$

Here x_i and u_i are states and controls respectively where x_0 is given. w_i are disturbances. Here we only consider a special case when the states, the controls and the disturbances are in one dimensional space. When the disturbances in the system are unknown, we usually model them as i.i.d. random variables W_i with given probability density function. In this case, the open-loop means the controls u_i do not depend on the disturbances [21] and we can further write the dynamic system in its random form:

$$X_{i+1} = \psi_i(X_i, u_i, W_i), \quad i = 0, \dots, d-1. \quad (80)$$

If we further define the vectors of states, controls and noises, i.e.,

$$X = (x_0, \dots, X_{d-1}), \quad u = (u_0, \dots, u_{d-1}), \quad W = (W_0, \dots, W_{d-1}), \quad (81)$$

then we can rewrite the dynamic system as

$$X = \Psi(X, u, W). \quad (82)$$

where Ψ is a function can be derived from ψ_i .

Our task now is to determine what is the 'best' control for the dynamic system (80) or (82) to minimize the expected cost

$$\mathbb{E}[\Phi(u, X)] \quad (83)$$

where Φ is a given function.

Here we focus on the case when X can be solved explicitly from the dynamic system (82), that is,

$$X = \xi(u, W). \quad (84)$$

In this case, the original problem can be reduced into the standard form of the stochastic optimization problem discussed in the paper, namely,

$$\min_{u \in U} \mathbb{E}[h(u, W)], \quad (85)$$

where

$$h(u, W) = \Phi(u, \xi(u, W)). \quad (86)$$

The integral form of the expected cost and its surrogate with N quadrature points are

$$\int_{\mathbb{R}^d} \Phi(u, \xi(u, w)) p(w) \, dW \approx \sum_{j=1}^N c_j \Phi(u, \xi(u, w_j)) p(w_j). \quad (87)$$

In order to illustrate the computational performance of our approach, we consider a classical example with linear dynamic system

$$X = AX + Bu + CW + x_0 \mathbf{e}_0. \quad (88)$$

and the quadratic objective function Φ

$$\Phi(u, X) = u^T P u + x^T Q x \quad (89)$$

where A, B, C, P and Q are given $d \times d$ matrices and x_0 is the given initial value. By solving (88), we get $\xi(u, W) = (I - A)^{-1}(Bu + CW + x_0 \mathbf{e}_0)$. Combining the expression of $\xi(u, W)$ with (89), we know that $h(u, W)$ is again a quadratic function.

The exact solution can be derived by using the certainty equivalence principle [1]. According to the principle, the solution of the stochastic control problem is the same as that of a corresponding deterministic problem when the objective function is quadratic and the constraints are linear. That means we can get the reference solution by numerically solving the deterministic problem (see appendix).

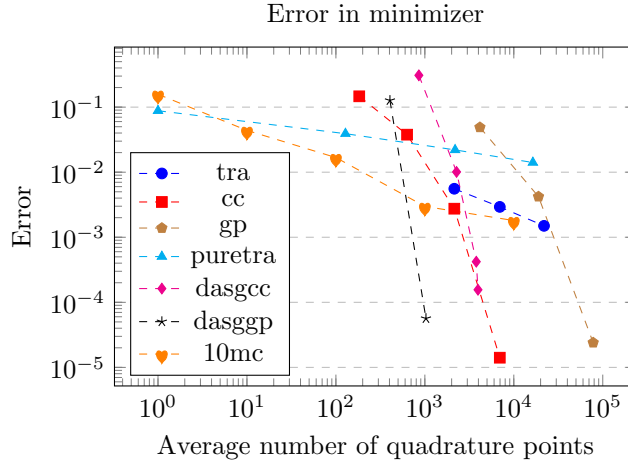
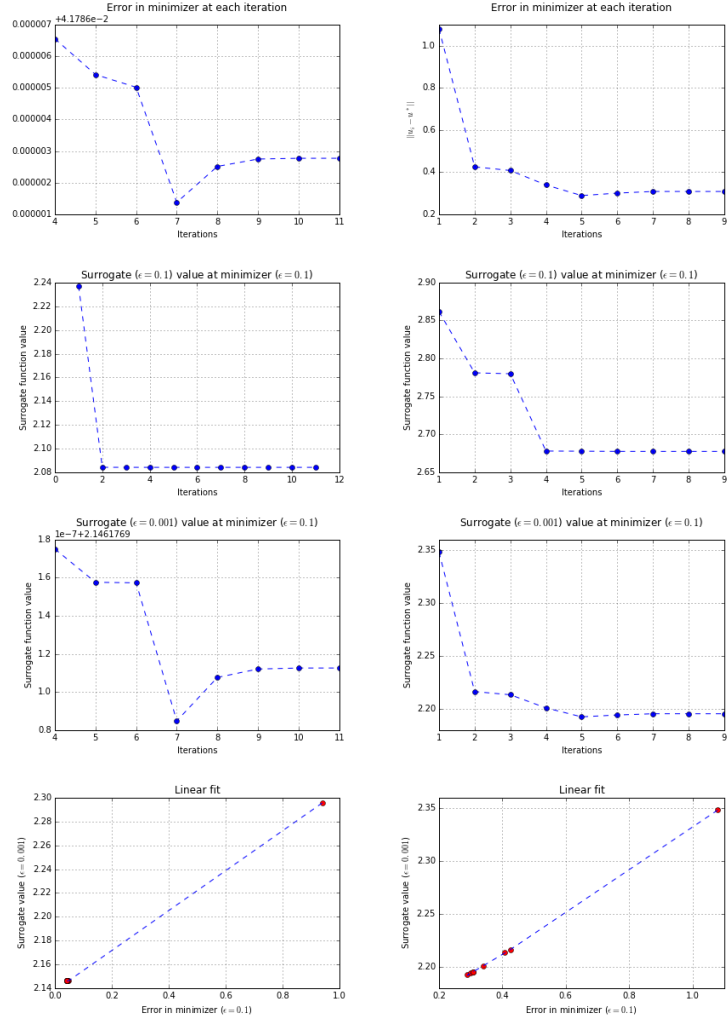


Figure 6: Compute the additive example with $d = 50$.

We test a 7 dimensional problem. We use an asymmetric distribution $\text{beta}(2, 3)$ here. We will get the exact solution with only rounding errors if we use a symmetric distribution. This is because the symmetric construction of the sparse grid will lead to the cancellation of the quadrature points pairs. We still use BFGS method as our optimization solver. The computational results are shown in the Figure 7.2. We compare the errors of the 9 different methods. They are product trapezoidal rule, the average of 10 runs Monte Carlo, three sparse grid method and three dimension adaptive sparse grid. We only record the data when sparse grid method and dimension adaptive sparse grid method start to converge. As can be seen from the figure, the dimension adaptive sparse grid methods converge faster than classical sparse grid methods for all three univariate rules. The results of sparse grid methods are much better than trapezoidal product rule and Monte Carlo method.

In Figure 7, we test our stopping criterion for the stochastic control problem with quadratic cost function and linear dynamic system. For both 6D and 7D examples, our method successfully predicts that we should stop at 7th iteration for 6D problem and 5th iteration for 7D problem.



(a) Clenshaw Curtis with $d = 6$

(b) Clenshaw Curtis with $d = 7$

Figure 7: Solve the stochastic control problem.

8 Conclusions

We apply the Newton-type methods in solving the stochastic optimisation problem and the dimension-adaptive sparse grid quadrature is used in approximating the integrals involved. In fact we can use more flexible discretisation scheme during the computation if we apply the OTDM. The dimension-adaptive sparse grid quadrature can effectively reduce the computational cost when we use it to compute an integral of which the dimensions are not equally important. When we applied it to solve the stochastic optimisation problem, we find it is more suitable to be used in the OTDM compared with the DTOM. This is because the OTDM allows us to choose 'best' downset in the dimension-adaptive sparse grid formula at each iteration and thus fully exploit the potential of the dimension-adaptive approach. The convergence of the OTDM can be make sure under the condition of Theorem (5.1). We give the condition when the convergence of our method might break down which leads to a rough stopping criterion. A good stopping criterion is crucial for reducing the computational cost when we solve high dimensional stochastic optimisation problems. We provide another more accurate and practical stopping criterion which only needs reasonable additional computation. We focus on the convex objective function in this paper. For non-convex problems, our approach can only find an approximated local minimizer. In order to solve more general stochastic optimisation problems, other solvers will be taken into consideration in the future research.

References

- [1] Dimitri P. Bertsekas. *Dynamic programming and optimal control. Vol. I. Third.* Athena Scientific, Belmont, MA, 2005, pp. xvi+543. ISBN: 1-886529-26-4.
- [2] Hans-Joachim Bungartz and Michael Griebel. “Sparse grids”. In: *Acta Numer.* 13 (2004), pp. 147–269. ISSN: 0962-4929. DOI: 10.1017/S0962492904000182. URL: <https://doi.org/10.1017/S0962492904000182>.
- [3] Michael Chen and Sanjay Mehrotra. *Epi-convergent scenario generation method for stochastic problems via sparse grid.* 2008.
- [4] Michael Chen, Sanjay Mehrotra, and Dávid Papp. “Scenario generation for stochastic optimization problems via the sparse grid method”. In: *Comput. Optim. Appl.* 62.3 (2015), pp. 669–692. ISSN: 0926-6003. DOI: 10.1007/s10589-015-9751-7. URL: <https://doi.org/10.1007/s10589-015-9751-7>.
- [5] Charles W. Clenshaw and Alan R. Curtis. “A method for numerical integration on an automatic computer”. In: *Numer. Math.* 2 (1960), pp. 197–205. ISSN: 0029-599X. DOI: 10.1007/BF01386223. URL: <https://doi.org/10.1007/BF01386223>.
- [6] Philip J. Davis and Philip Rabinowitz. *Methods of numerical integration.* Second. Computer Science and Applied Mathematics. Academic Press Inc., Orlando, FL, 1984, pp. xiv+612. ISBN: 0-12-206360-0.
- [7] J. E. Dennis Jr. and Homer F. Walker. “Inaccuracy in quasi-Newton methods: local improvement theorems”. In: *Math. Programming Stud.* 22 (1984). Mathematical programming at Oberwolfach, II (Oberwolfach, 1983), pp. 70–85. ISSN: 0303-3929. DOI: 10.1007/bfb0121009. URL: <https://doi.org/10.1007/bfb0121009>.
- [8] Josef Dick, Frances Y Kuo, and Ian H Sloan. “High-dimensional integration: the quasi-Monte Carlo way”. In: *Acta Numerica* 22 (2013), p. 133.
- [9] Jochen Garcke. “Sparse grids in a nutshell”. In: *Sparse grids and applications.* Vol. 88. Lect. Notes Comput. Sci. Eng. Springer, Heidelberg, 2013, pp. 57–80. DOI: 10.1007/978-3-642-31703-3. URL: <https://doi.org/10.1007/978-3-642-31703-3>.
- [10] T. Gerstner and M. Griebel. “Dimension-adaptive tensor-product quadrature”. In: *Computing* 71.1 (2003), pp. 65–87. ISSN: 0010-485X. DOI: 10.1007/s00607-003-0015-5. URL: <https://doi.org/10.1007/s00607-003-0015-5>.
- [11] Thomas Gerstner and Michael Griebel. “Numerical integration using sparse grids”. In: *Numer. Algorithms* 18.3-4 (1998), pp. 209–232. ISSN: 1017-1398. DOI: 10.1023/A:1019129717644. URL: <https://doi.org/10.1023/A:1019129717644>.
- [12] Brendan Harding. “Fault Tolerant Computation of Hyperbolic Partial Differential Equations with the Sparse Grid Combination Technique”. PhD thesis. The Australian National University, 2016.
- [13] Markus Hegland. “Adaptive sparse grids”. In: *Anziam Journal* 44 (2003), pp. 335–353.

- [14] Markus Hegland et al. “Recent developments in the theory and application of the sparse grid combination technique”. In: *Software for exascale computing—SPPEXA 2013–2015*. Vol. 113. Lect. Notes Comput. Sci. Eng. Springer, [Cham], 2016, pp. 143–163.
- [15] Markus Holtz. “Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance”. PhD thesis. university of Bonn, 2008.
- [16] C. T. Kelley. *Iterative methods for linear and nonlinear equations*. Vol. 16. Frontiers in Applied Mathematics. With separately available software. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1995, pp. iv+165. ISBN: 0-89871-352-8. DOI: 10.1137/1.9781611970944. URL: <https://doi.org/10.1137/1.9781611970944>.
- [17] C. T. Kelley. *Solving nonlinear equations with Newton’s method*. Vol. 1. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003, pp. xiv+104. ISBN: 0-89871-546-6. DOI: 10.1137/1.9780898718898. URL: <https://doi.org/10.1137/1.9780898718898>.
- [18] Nicholas Metropolis and Stanislaw Ulam. “The monte carlo method”. In: *Journal of the American statistical association* 44.247 (1949), pp. 335–341.
- [19] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*. SIAM, 1992.
- [20] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Second. Springer Series in Operations Research and Financial Engineering. Springer, New York, 2006, pp. xxii+664. ISBN: 978-0387-30303-1; 0-387-30303-0.
- [21] Bernt Øksendal. *Stochastic differential equations*. Fifth. Universitext. An introduction with applications. Springer-Verlag, Berlin, 1998, pp. xx+324. ISBN: 3-540-63720-6. DOI: 10.1007/978-3-662-03620-4. URL: <https://doi.org/10.1007/978-3-662-03620-4>.
- [22] T. N. L. Patterson. “The optimum addition of points to quadrature formulae”. In: *Math. Comp.* 22 (1968), 847–856; addendum, *ibid.* 22.104, loose microfiche supp. (1968), pp. C1–C11. ISSN: 0025-5718. DOI: 10.2307/2004583. URL: <https://doi.org/10.2307/2004583>.
- [23] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. Vol. 10. John Wiley & Sons, 2016.
- [24] Ian H Sloan and Henryk Woźniakowski. “When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?” In: *Journal of Complexity* 14.1 (1998), pp. 1–33.
- [25] Tjalling J. Ypma. “The effect of rounding errors on Newton-like methods”. In: *IMA Journal of Numerical Analysis* 3.1 (1983), pp. 109–118.
- [26] Yuancheng Zhou and Markus Hegland. “The application of sparse grid quadrature in solving stochastic optimisation problems”. In: *ANZIAM Journal* 60 (2018), pp. 16–32.