

Grieshammer, Max; Pflug, Lukas; Stingl, Michael; Uihlein, Andrian

Article — Published Version

The continuous stochastic gradient method: part I–convergence theory

Computational Optimization and Applications

Provided in Cooperation with:

Springer Nature

Suggested Citation: Grieshammer, Max; Pflug, Lukas; Stingl, Michael; Uihlein, Andrian (2023) : The continuous stochastic gradient method: part I–convergence theory, Computational Optimization and Applications, ISSN 1573-2894, Springer US, New York, NY, Vol. 87, Iss. 3, pp. 935-976, <https://doi.org/10.1007/s10589-023-00542-8>

This Version is available at:

<https://hdl.handle.net/10419/312466>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.


If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



<https://creativecommons.org/licenses/by/4.0/>



The continuous stochastic gradient method: part I—convergence theory

Max Grieshammer¹ · Lukas Pflug^{1,2} · Michael Stingl¹ · Andrian Uihlein¹ 

Received: 12 January 2023 / Accepted: 31 October 2023 / Published online: 23 November 2023
© The Author(s) 2023

Abstract

In this contribution, we present a full overview of the *continuous stochastic gradient* (CSG) method, including convergence results, step size rules and algorithmic insights. We consider optimization problems in which the objective function requires some form of integration, e.g., expected values. Since approximating the integration by a fixed quadrature rule can introduce artificial local solutions into the problem while simultaneously raising the computational effort, stochastic optimization schemes have become increasingly popular in such contexts. However, known stochastic gradient type methods are typically limited to expected risk functions and inherently require many iterations. The latter is particularly problematic, if the evaluation of the cost function involves solving multiple state equations, given, e.g., in form of partial differential equations. To overcome these drawbacks, a recent article introduced the CSG method, which reuses old gradient sample information via the calculation of design dependent integration weights to obtain a better approximation to the full gradient. While in the original CSG paper convergence of a subsequence was established for a diminishing step size, here, we provide a complete convergence analysis of CSG for constant step sizes and an Armijo-type line search. Moreover, new methods to obtain the integration weights are presented, extending the application range of CSG to problems involving higher dimensional integrals and distributed data.

✉ Andrian Uihlein
andrian.uihlein@fau.de

Max Grieshammer
max.grieshammer@fau.de

Lukas Pflug
lukas.pflug@fau.de

Michael Stingl
michael.stingl@fau.de

¹ Department of Mathematics, Chair of Applied Mathematics, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

² FAU Competence Center Scientific Computing, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany

Keywords Stochastic gradient scheme · Convergence analysis · Step size rule · Backtracking line search · Constant step size

Mathematics Subject Classification 65K05 · 90C06 · 90C15 · 90C30

1 Introduction

In this contribution, we present a full overview of the *continuous stochastic gradient* (CSG) method, including convergence results, step size rules, algorithmic insights and applications from topology optimization. The prototype idea for CSG was first proposed in [1]. Therein, it was shown that for expected-valued objective functions, CSG with diminishing step sizes and exact integration weights (see Sect. 3) almost surely produces a subsequence converging to a stationary point [1, Theorem 20].

This work generalizes this result, providing proofs for almost sure convergence of the full sequence of iterates in the case of constant step sizes (Theorems 5.1 and 5.2) and backtracking line search techniques (Theorem 6.1). Additionally, the convergence results hold in a less restrictive setting and for generalized approaches to the integration weight calculation (see Sect. 3). Before going into details, we want to explain the reason for introducing what seems like yet another first-order stochastic optimization scheme.

1.1 Motivation

Within PDE-constrained optimization, settings with expected-valued objective functions arise in numerous applications, ranging from purely stochastic settings, like machine learning or noisy simulations [2, 3], to fully deterministic problems, in which one is interested in a design that is optimal for an infinite set of outer parameters, e.g. [4–7]. In this work, we assume the stochasticity to appear solely in the objective function. For problems in which the underlying state equation itself includes uncertainties, different techniques have been proposed in the past [8–11]. Moreover, we restrict our convergence analysis to the finite-dimensional case, and leave a generalization to infinite-dimensional spaces, see, e.g., [12, 13], for future work. Especially in large scale settings, one usually does not consider deterministic approaches (see, e.g., [14, 15]) for the solution of such problems, as they are generally too computationally expensive or even intractable. Instead, one uses stochastic optimization schemes, like the *Stochastic Gradient* (SG) [16] or *Stochastic Average Gradient* (SAG) method [17]. A large number of schemes have been derived from these and thoroughly analyzed, including *sequential quadratic programming for stochastic optimization* [18, 19], *quasi-Newton stochastic gradient schemes* [20–23] and the well known adaptive gradient schemes *Adam* & *AdaGrad* [24, 25], to name some prominent examples.

Problematically, such methods rely on a heavily restrictive setting, in which the objective function value of a design u is simply given as the expected value of some quantity j , i.e., $J(u) = \mathbb{E}_x[j(u, x)]$. Even the basic setting of nesting two expectation values, i.e., $J(u) = \mathbb{E}_y[j_2(y, \mathbb{E}_x[(j_1(u, x))])]$, is beyond the scope of the mentioned schemes and requires special techniques, e.g. the *Stochastic Composition Gradient*

Descent (SCGD) method [26], which itself is again only applicable in this specific setting.

Problems with such complex structures arise in several applications, see, e.g., [27], where we investigate an application from the field of optimal nanoparticle design. Therein, our main interest lies in the optical properties of nanoparticles. Specifically, the color of particulate products, which has been of great interest for many fields of research [28–33], is what we are trying to optimize for. This and similar applications serve only as motivation in this paper. However, in the numerical analysis of CSG [27], it is demonstrated that CSG indeed represents an efficient approach to such problems. While a detailed introduction to this setting is given in [27], we want to briefly summarize the problems arising in this application.

To obtain the color of a particulate product, we need to calculate the important optical properties of the nanoparticles in the product. For each particle design, this requires solving the time-harmonic Maxwell's equations, which, depending of the setting, is numerically expensive. Furthermore, the color of the whole product is not determined by the optical properties of a single particle. Instead, we need to average these properties over, e.g., the particle design distribution and their orientation in the particulate product. Afterwards, the averaged values are used to calculate intermediate results for the special setting. These results then need to be integrated over the range of wavelengths, which are visible to the human eye, to obtain the color of the particulate product. Finally, the objective function uses the resulting color in a nonlinear fashion, before yielding the actual objective function value. In general, the objective function has the form

$$J(u) = j_3(u, \mathbb{E}_y[j_2(u, y, \mathbb{E}_x[j_1(u, x, y)])])$$

and can easily involve even more convoluted terms in more advanced settings.

On the one hand, the computational cost of deterministic approaches to such problems range from tremendous to infeasible, since j_1 is typically not easy to evaluate. On the other hand, standard schemes from stochastic optimization, like the ones mentioned above, simply cannot solve the full problem. Thus, we are in the need for a general method to tackle optimization problems, which are given by arbitrary concatenations of nonlinear functions and expectation values.

1.2 Properties of CSG

As mentioned in the previous section, the CSG method aims to offer a new approach to optimization problems that involve integration of properties, which are expensive to evaluate. In each iteration, CSG draws a very small number (typically 1) of random samples, as is the case for SG. However, instead of discarding the information collected through these evaluations after the iteration, these results are stored. In later iterations, all of the information collected along the way is used to build an approximation to the full objective function and its gradient, by a special linear combination. The weights appearing in this linear combination, which we call *integration weights*, can be calculated in several different fashions, which are detailed in Sect. 3.

This approach of approximating an unknown function by old sample information is similar to surrogate methods from Bayesian optimization [34–36], but differs from

these techniques in an important aspect: While surrogate models require a rather low *combined* dimension of design and integration, the performance of CSG is only impacted by the dimension of integration. This is demonstrated in [27], where a problem with high dimensional design is optimized subject to a rather low dimensional integration.

As a key result for CSG, we are able to show that the approximation error in both the gradient and the objective function approximation vanishes during the optimization process (Lemma 4.6). Thus, in the course of the iterations, CSG more and more behaves like an exact full gradient algorithm and is able to solve optimization problems far beyond the scope of standard SG-type methods. Furthermore, we show that this special behaviour results in CSG having convergence properties similar to full gradient descent methods, while keeping advantages of stochastic optimization approaches, i.e., each step is computationally cheap. To be precise, our main results consist of proving convergence to a stationary point for constant step sizes (Theorem 5.1) and an Armijo-type line search (Theorem 6.1), which is based on the gradient and objective function approximations of CSG.

1.3 Limitations of the method

While CSG combines advantages of deterministic and stochastic optimization schemes, the hybrid approach also yields drawbacks, which we try to address throughout this contribution. As mentioned earlier, the intended application for CSG lies in expected-valued optimization problems, in which solving the state problem is computationally expensive. In many other settings that heavily rely on stochastic optimization methods, e.g., neural networks, the situation is different. Here, we can efficiently obtain a stochastic (sub-)gradient, meaning that we are better off simply performing millions of SG iterations, than a few thousand CSG steps. In these situations, the inherent additional computational effort that lies within the calculation of the CSG integration weights (see Sect. 3) is no longer negligible and usually can not be compensated by the improved gradient approximation.

Furthermore, while the design dimension has almost no impact on the performance, the convergence rate of CSG worsens as the dimension of integration increases. While this can be avoided, if the objective function imposes additional structure, it remains a drawback in general. A detailed analysis of this issue can be found in [27].

We emphasize that CSG and SG-type methods share many similarities. However, their intended applications are complementary to each other, with SG preferring objective functions of simple structure and computationally cheap sampling, while CSG prefers the opposite.

1.4 Structure of the paper

Section 2 states the general framework of this contribution as well as the basic assumptions we impose throughout the paper. Furthermore, the general CSG method is presented.

The integration weights, which play an important role in the CSG scheme, are detailed in Sect. 3. Therein, we introduce four different methods to obtain weights

which satisfy the necessary assumptions made in Sect. 2 and analyze some of their properties. The section also describes techniques to implement mini-batches in the CSG method.

Auxiliary results concerning CSG are given in Sect. 4. This includes the important gradient approximation property of CSG (Lemma 4.6) and details on how to generalize CSG to even broader settings (Remark 4.1). The first main result, i.e., convergence for constant steps (Theorem 5.1 and Theorem 5.2), is presented in Sect. 5.

Afterwards, in Sect. 6, we incorporate an Armijo-type line search in the CSG method and provide a convergence analysis for the associated optimization scheme (Theorem 6.1). Furthermore, we introduce heuristics to efficiently approximate some hyperparameters. The resulting SCIBL-CSG method (Algorithm 4) achieves identical convergence results, but does not require any step sizes tuning.

The key features of CSG are numerically demonstrated for academic examples in Sect. 7. A detailed numerical analysis of CSG, concerning the performance for non-academic examples and convergence rates, is not part of this contribution, as this can be found in [27].

2 Setting and assumptions

In this section, we introduce the general setting as well as formulate and motivate the assumptions made throughout this contribution. Additionally, the basic CSG algorithm is presented.

2.1 Setting

As mentioned above, the convergence analysis of the CSG method is carried out in a simplified setting, where the objective function is given by a single expected value. For the general case, see Remark 4.1.

Since our convergence analysis will heavily rely on the similarities between CSG and a full gradient scheme, we adapt the standard deterministic setting of an L -smooth objective function (Remark 2.3). Moreover, as the CSG method approximates gradients by a nearest neighbor model, we require all appearing functions to be Lipschitz continuous with respect to all arguments (Assumption 2.3).

Definition 2.1 (*Objective Function*) For $d_o, d_r \in \mathbb{N}$, we introduce the set of admissible optimization variables $\mathcal{U} \subseteq \mathbb{R}^{d_o}$ and the parameter set $\mathcal{X} \subseteq \mathbb{R}^{d_r}$. The objective function $J : \mathcal{U} \rightarrow \mathbb{R}$ is defined as

$$J(u) := \mathbb{E}[j(u, X)] = \int_{\mathcal{X}} j(u, x) \mu(dx),$$

where we assume $j \in C^1(\mathcal{U} \times \mathcal{X}; \mathbb{R})$ to be measurable and $X \sim \mu$. The (simplified) optimization problem is then given by

$$\min_{u \in \mathcal{U}} \int_{\mathcal{X}} j(u, x) \mu(dx). \quad (1)$$

Since \mathcal{U} and \mathcal{X} are finite-dimensional, we do not have to consider specific norms on these spaces due to equivalence of norms and can instead choose them problem specific. In the following, we will denote them simply by $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{X}}$.

During the optimization, we need to draw independent random samples of the random variable X , as stated in the following assumption.

Assumption 2.1 (*Sample Sequence*) The sequence of samples $(x_n)_{n \in \mathbb{N}}$ is a sequence of independent identically distributed realizations of the random variable $X \sim \mu$.

Remark 2.1 (*Almost Sure Density*) We define the support of the measure μ as the (closed) subset

$$\text{supp}(\mu) := \{x \in \mathcal{X} : \mu(\mathcal{B}_\varepsilon(x)) > 0 \text{ for all } \varepsilon > 0\} \subseteq \mathbb{R}^{d_r}.$$

It is important to note, that a sample sequence satisfying Assumption 2.1 is dense in $\text{supp}(\mu)$ with probability 1. This can be seen as follows:

Let $x \in \text{supp}(\mu)$ and $\varepsilon > 0$. Then, given an independent identically distributed sequence $x_1, x_2, \dots \sim \mu$, we have

$$\mathbb{P}(x_n \notin \mathcal{B}_\varepsilon(x)) = 1 - \mu(\mathcal{B}_\varepsilon(x)) < 1.$$

Hence, by the Borel–Cantelli Lemma [37, Theorem 2.7],

$$\mathbb{P}(x_n \notin \mathcal{B}_\varepsilon(x) \text{ for all } n \in \mathbb{N}) = 0.$$

Thus, the sequence $(x_n)_{n \in \mathbb{N}}$ is dense in $\text{supp}(\mu)$ with probability 1.

Remark 2.2 (*Almost Sure Convergence Results*) The (almost sure) density of the sample sequence plays a crucial role in the upcoming proofs. Hence, the convergence results presented in this contribution all hold in the almost sure sense. However, to improve the readability, we refrain from always mentioning this explicitly.

Moreover, the sets \mathcal{U} and \mathcal{X} need to satisfy additional regularity conditions. Since CSG handles constraints by projecting steps onto the set of admissible designs \mathcal{U} , we need to ensure this operation is well-defined. Furthermore, the approximation property (Lemma 4.6) of the nearest neighbor model in CSG is fundamentally based on the density of sample points $(x_n)_{n \in \mathbb{N}}$, which is why we require $\text{supp}(\mu)$ to be bounded.

Assumption 2.2 (*Regularity of \mathcal{U} , \mathcal{X} and μ*) The set $\mathcal{U} \subseteq \mathbb{R}^{d_o}$ is compact and convex. The set $\mathcal{X} \subseteq \mathbb{R}^{d_r}$ is bounded with $\text{supp}(\mu) \subseteq \mathcal{X}$.

Notice that the second part of Assumption 2.2 can always be achieved, as long as $\text{supp}(\mu) \subseteq \mathbb{R}^{d_r}$ is bounded.

Finally, as in the deterministic case, we assume the gradient of the objective function to be Lipschitz continuous, in order to obtain convergence for constant step sizes.

Assumption 2.3 (*Regularity of j*) The function $\nabla_1 j : \mathcal{U} \times \mathcal{X} \rightarrow \mathbb{R}^{d_o}$ is bounded and Lipschitz continuous, i.e., there exists constants $C, L_j \in \mathbb{R}_{>0}$ such that

$$\|\nabla_1 j(u, x)\| \leq C,$$

$$\|\nabla_1 j(u_1, x_1) - \nabla_1 j(u_2, x_2)\| \leq L_j (\|u_1 - u_2\|_{\mathcal{U}} + \|x_1 - x_2\|_{\mathcal{X}})$$

for all $u, u_1, u_2 \in \mathcal{U}$ and $x, x_1, x_2 \in \mathcal{X}$.

Remark 2.3 (Regularity of ∇J) By Assumptions 2.1 to 2.3, $J : \mathcal{U} \rightarrow \mathbb{R}$ is L -smooth, i.e., there exists $L > 0$ such that

$$\|\nabla J(u_1) - \nabla J(u_2)\| \leq L\|u_1 - u_2\|_{\mathcal{U}} \quad \text{for all } u_1, u_2 \in \mathcal{U}.$$

2.2 The CSG method

Given a starting point $u_1 \in \mathcal{U}$ and a random parameter sequence x_1, x_2, \dots according to Assumption 2.1, the basic CSG method is stated in Algorithm 1. In each iteration, the inner objective function j and gradient $\nabla_1 j$ are evaluated only at the current design-parameter-pair (u_n, x_n) . Afterwards, the integrals appearing in J and ∇J are approximated by a linear combination, consisting of all samples accumulated in previous iterations.

The coefficients $(\alpha_k)_{k=1, \dots, n}$ appearing in Step 4 of Algorithm 1, which we call *integration weights*, are what differentiates CSG from other stochastic optimization schemes. In [1], where the main idea of the CSG method was proposed for the first time, a special choice of how to calculate these weights was already presented. A recap of this procedure as well as several new methods to obtain the integration weights is given in detail in Sect. 3.

Furthermore, $\mathcal{P}_{\mathcal{U}}$ appearing in Step 8 of Algorithm 1 denotes the orthogonal projection (in the sense of $\|\cdot\|_{\mathcal{U}}$), i.e.,

$$\mathcal{P}_{\mathcal{U}}(u) := \arg \min_{\bar{u} \in \mathcal{U}} \|u - \bar{u}\|_{\mathcal{U}}.$$

The final general assumption we will use throughout this contribution is related to the integration weights mentioned above.

Assumption 2.4 (Integration Weights) Denote the sequence of designs and random parameters generated by Algorithm 1 until iteration $n \in \mathbb{N}$ by $(u_k)_{k=1, \dots, n}$ and $(x_k)_{k=1, \dots, n}$, respectively. Define $k^n : \mathcal{X} \rightarrow \{1, \dots, n\}$ as

$$k^n(x) := \arg \min_{k=1, \dots, n} (\|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}}). \quad (2)$$

We assume the integration weights appearing in Algorithm 1 satisfy the following:

For all $n \in \mathbb{N}$, there exists a probability measure μ_n on \mathcal{X} such that

- (a) The integration weights $(\alpha_k^{(n)})_{k=1, \dots, n}$ in iteration $n \in \mathbb{N}$ are given by

$$\alpha_k^{(n)} = \int_{\mathcal{X}} \delta_{k^n(x)}(k) \mu_n(dx) \quad \text{for all } k = 1, \dots, n, \quad (3)$$

where $\delta_{k^n(x)}(k)$ corresponds to the Dirac measure of $k^n(x)$.

(b) The measures $(\mu_n)_{n \in \mathbb{N}}$ converge weakly to μ , i.e., $\mu_n \Rightarrow \mu$ for $n \rightarrow \infty$, where

$$\mu_n \Rightarrow \mu \quad \text{iff} \quad \int_{\mathcal{X}} f(x) \mu_n(dx) \rightarrow \int_{\mathcal{X}} f(x) \mu(dx) \quad \text{for all } f \in \mathcal{C}^{0,1}(\mathcal{X}, \mathbb{R}).$$

There is a very simple idea hidden behind the technicalities of Assumption 2.4. Condition (3) states that the integration weights should be somehow based on a nearest neighbor approximation

$$\nabla_1 j(u_n, x) \approx \nabla_1 j(u_{k^n(x)}, x_{k^n(x)}),$$

while the condition $\mu_n \Rightarrow \mu$ ensures that the weight of a sample is reasonably chosen, i.e.,

$$\alpha_k^{(n)} = \int_{\mathcal{X}} \delta_{k^n(x)}(k) \mu_n(dx) \approx \int_{\mathcal{X}} \delta_{k^n(x)}(k) \mu(dx).$$

Remark 2.4 (*Choice of Neighbor*) For some $x \in \mathcal{X}$, there might not exist a *unique* nearest neighboring sample to (u_n, x) . As a result, $k^n(x)$, as defined in (2), may consist of more than one index, resulting in x to be counted towards multiple integration weights in (3). Thus, if $k^n(x)$ is not unique, we have to pick one of the nearest neighbors and neglect the other possible choices. However, it is not important how we make that choice, so we could, for example, replace (2) by

$$\tilde{k}^n(x) := \min \left\{ \arg \min_{k=1, \dots, n} (\|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}}) \right\},$$

or choose an index at random in these special instances. Thus, in the remainder of this work, we refrain from imposing a specific choice of neighbors and use the notationally reduced formulation (2), essentially assuming that the set of $x \in \mathcal{X}$ with more than one nearest neighbor to (u_n, x) is of μ_n -measure zero.

Due to the finite-dimensional setting, all convergence results proven in this contribution hold independent of the chosen norm on $\mathcal{U} \times \mathcal{X}$ implied by (2). However, the specific choice may strongly influence the behavior (see, Sect. 3.5) and performance of CSG. Further insight on the integration weights and multiple methods to obtain weights satisfying Assumption 2.4 are given in the following Sect. 3.

3 Integration weights

In this section, we present four methods on how to obtain integration weights satisfying Assumption 2.4 in practice. The methods differ greatly in their associated computational cost and accuracy, allowing us to make an appropriate choice in different settings. Moreover, two of the proposed techniques require no knowledge about the underlying measure μ , allowing us to use them in purely data-driven applications.

Algorithm 1 General CSG Method

```

while Termination condition not met do
  Sample objective function (optional):
     $j_n = j(u_n, x_n)$ 
  Sample gradient:
     $g_n = \nabla_1 j(u_n, x_n)$ 
  Calculate integration weights:
     $\alpha_k$ 
  Calculate search direction:
     $\hat{G}_n = \sum_{k=1}^n \alpha_k g_k$ 
  Compute objective function value approximation (optional):
     $\hat{J}_n = \sum_{k=1}^n \alpha_k j_k$ 
  Choose step size:
     $\tau_n$ 
  Gradient step:
     $u_{n+1} = \mathcal{P}_{\mathcal{U}}(u_n - \tau_n \hat{G}_n)$ 
  Update index:
     $n \leftarrow n + 1$ 
end while

```

The general CSG method as proposed in [1] for the simplified setting (1). Further information on how to carry out the integration weight calculation in practice is given in Section 3.

We start with a brief motivation: Suppose that we are currently in the 9th iteration of the CSG algorithm. So far, we have sampled $\nabla_1 j(u, x)$ at nine points $(u_i, x_i)_{i=1, \dots, 9}$ and the full gradient at the current design is given by

$$\nabla J(u_9) = \int_{\mathcal{X}} \nabla_1 j(u_9, x) \mu(dx).$$

In Fig. 1, the situation is shown for the case $d_r = d_o = 1$. How can we use the samples $(\nabla_1 j(u_i, x_i))_{i=1, \dots, 9}$ in an optimal fashion, to build an approximation to $\nabla J(u_9)$?

3.1 Exact weights

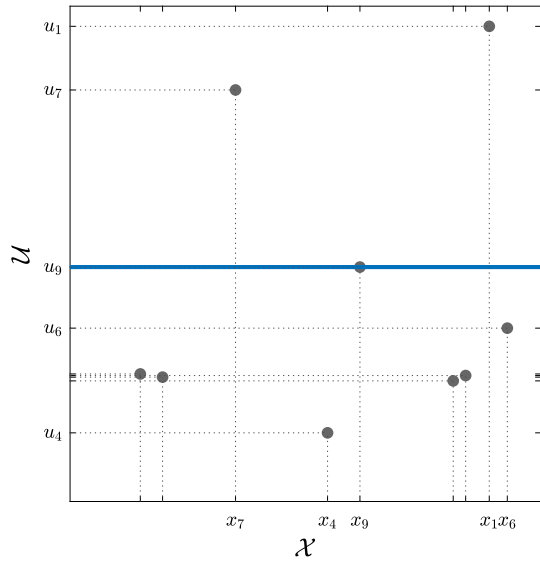
To approximate the value of the integral along the bold line, we may use a nearest neighbor approximation. The underlying idea is visualized in Fig. 2. Thus, we define the sets

$$M_k := \{x \in \mathcal{X} : \|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}} < \|u_n - u_j\|_{\mathcal{U}} + \|x - x_j\|_{\mathcal{X}} \text{ for all } j \in \{1, \dots, n\} \setminus \{k\}\}.$$

By construction, M_k contains all points $x \in \mathcal{X}$, such that (u_n, x) is closer to (u_k, x_k) than to any other previous point we evaluated $\nabla_1 j$ at. The full gradient is then approximated in a piecewise constant fashion by

$$\nabla J(u_n) = \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx)$$

Fig. 1 The grey dots represent our nine sample points $(u_i, x_i) \in \mathcal{U} \times \mathcal{X}$. The full gradient of the objective function is obtained by integrating $\nabla_1 j(u_9, \cdot)$ along the blue line (Color figure online)



$$\begin{aligned}
 &= \sum_{k=1}^n \int_{M_k} \nabla_1 j(u_n, x) \mu(dx) \\
 &\approx \sum_{k=1}^n \nabla_1 j(u_k, x_k) \mu(M_k).
 \end{aligned}$$

Therefore, we call $\alpha_k^{\text{ex}} = \mu(M_k)$ *exact* integration weights, since they are based on an exact nearest neighbor approximation. These weights were first introduced in [1] and offer the best approximation to the full gradient. However, the calculation of the exact integration weights requires full knowledge of the measure μ and is based on a Voronoi tessellation [38], which is computationally expensive for high dimensions of $\mathcal{U} \times \mathcal{X}$.

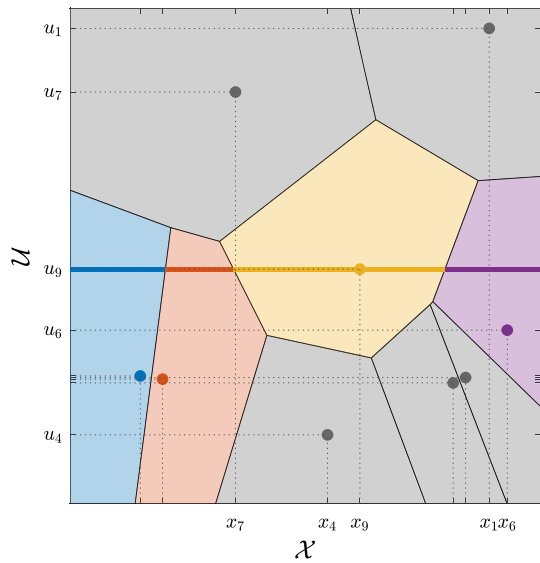
Note that, in the special case $\dim(\mathcal{X}) = 1$, the calculation of the tessellation can be circumvented, regardless of $\dim(\mathcal{U})$. Instead, the intersection points of the line $\{u_n\} \times \mathcal{X}$ and all faces of active Voronoi cells can be obtained directly by solving the equations appearing in the definition of M_k . This, however, still requires us to solve $\mathcal{O}(n^2)$ quadratic equations per CSG iteration.

3.2 Exact hybrid weights

In some settings, the dimension of \mathcal{X} can be very small compared to the dimension of \mathcal{U} . Hence, we might avoid computing a Voronoi tessellation in $\mathcal{U} \times \mathcal{X}$ by treating these spaces separately. For this, we introduce the sets

$$\tilde{M}_i := \{x \in \mathcal{X} : \|x - x_i\|_{\mathcal{X}} < \|x - x_j\|_{\mathcal{X}} \text{ for all } j \in \{1, \dots, n\} \setminus \{i\}\}.$$

Fig. 2 For the exact integration weights, α_i represents to the measure of the line segment that lies in the Voronoi cell around (u_i, x_i) . The grey cells correspond to samples with integration weight 0



Denoting by 1_{M_k} the indicator function of M_k , we define the *exact hybrid* weights as

$$\alpha_k^{\text{eh}} = \sum_{i=1}^n 1_{M_k}(x_i) \mu(\tilde{M}_i).$$

Notice that the sets M_k still appear in the definition of the exact hybrid weights, but do not need to be calculated explicitly. Instead, we only have to find the nearest sample point to $(u_n, x_i)_{i=1, \dots, n}$, which can be done efficiently even in large dimensions. We do, however, still require knowledge of μ . The idea of the exact hybrid weights is captured in Fig. 3.

3.3 Inexact hybrid weights

To calculate the integration weights in the case that the measure μ is unknown, we may approximate $\mu(\tilde{M}_i)$ empirically. All that is required for this approach is additional samples of the random variable X . To be precise, we draw enough samples such that in iteration n , we have in total $f(n)$ samples of X , where $f: \mathbb{N} \rightarrow \mathbb{N}$ is a function which is strictly increasing and satisfies $f(n) \geq n$ for all $n \in \mathbb{N}$. It is important to note, that we still evaluate $\nabla_1 j(u_n, \cdot)$ at only one of these points, which we denote by x_{j_n} . Thus, exchanging $\mu(\tilde{M}_i)$ by its empirical approximation yields the *inexact hybrid* weights

$$\alpha_k^{\text{ih}} = \frac{1}{f(n)} \sum_{i=1}^n 1_{M_k}(x_{j_i}) \sum_{m=1}^{f(n)} 1_{\tilde{M}_{j_i}}(x_m).$$

Fig. 3 The red stars on the bold line represent the points (u_n, x_i) . For each cell, we determine the number of these points inside the cell and combine their corresponding measures $\mu(\tilde{M}_i)$, indicated by the colored line segments. Since $\dim(\mathcal{X}) = 1$ in the shown example, the sets \tilde{M}_i are simply intervals, where the endpoints are given by the midpoints of two neighboring x_i

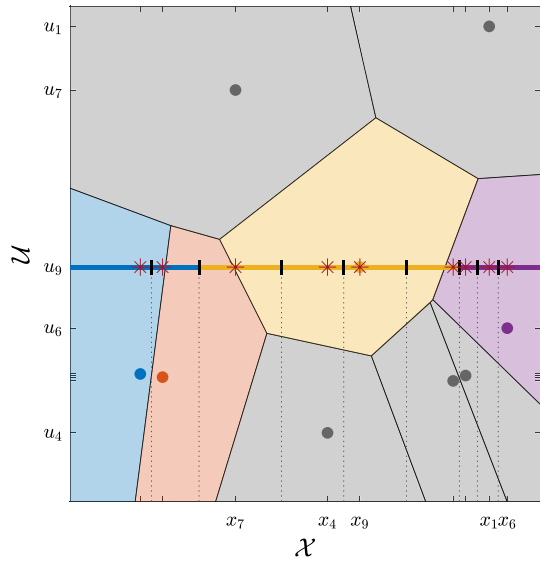
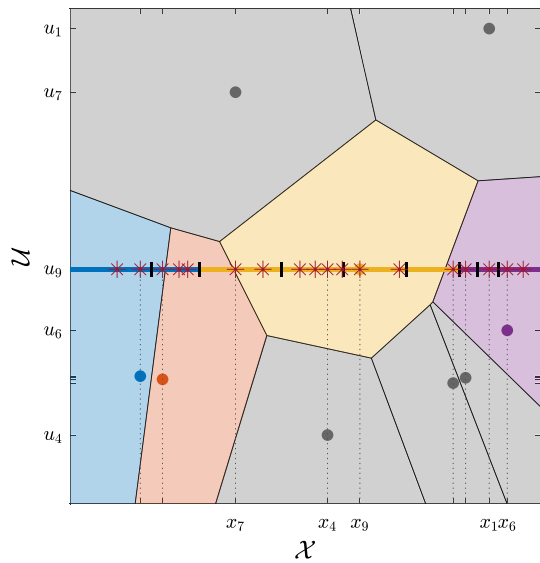
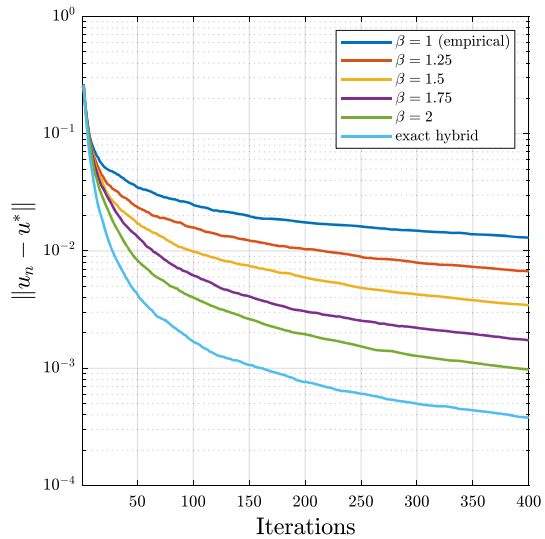


Fig. 4 The red stars on the bold line represent the points $(u_n, x_i)_{i=1, \dots, f(n)}$. Similar to the exact hybrid weights, for each cell of the Voronoi diagram, we count the points $(u_n, x_{j_i})_{i=1, \dots, n}$ inside the cell (marked by dashed lines). Instead of assigning them the weights $\mu(\tilde{M}_{j_i})$, we weight them by an empirical approximation to this quantity, i.e., by the percentage of random samples $(x_i)_{i=1, \dots, f(n)}$ that lie in \tilde{M}_{j_i}



How fast f grows determines not only the quality of the approximation, but also the computational complexity of the weight calculation. Based on the choice of f , the inexact hybrid weights interpolate between the exact hybrid weights and the empirical weights, which will be introduced below. In Fig. 5, this behavior is shown for functions of the form $f(n) = \lfloor n^\beta \rfloor$ with $\beta \geq 1$. Figure 4 illustrates the general concept of the inexact hybrid weights.

Fig. 5 Median of the results for problem (20), initialized with 1000 random starting points, for empirical weights, exact hybrid weights and several inexact hybrid weights. The function f appearing in the inexact hybrid weights was chosen as $f(n) = \lfloor n^\beta \rfloor$



3.4 Empirical weights

The *empirical* weights offer a computationally cheap alternative to the methods mentioned above. Their calculation does not require any knowledge of the measure μ . For the empirical weights, the quantity $\mu(M_k)$ is directly approximated by its empirical counterpart, i.e.,

$$\alpha_k^e = \frac{1}{n} \sum_{i=1}^n 1_{M_k}(x_i).$$

The corresponding picture is shown in Fig. 6. By iteratively storing the distances $\|x_i - x_j\|$, $i, j < n$, the empirical weights can be calculated very efficiently.

3.5 Metric on $\mathcal{U} \times \mathcal{X}$

As mentioned after Assumption 2.4, the convergence results proven in this contribution are independent of the specific inner norms on \mathcal{U} and \mathcal{X} , denoted by $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{X}}$. Furthermore, they also hold if we substitute the outer norm on $\mathcal{U} \times \mathcal{X}$, appearing in (2), e.g., by a generalized L^1 -outer norm, i.e.,

$$\|(u, x)\|_{\mathcal{U} \times \mathcal{X}} = c_1 \|u\|_{\mathcal{U}} + c_2 \|x\|_{\mathcal{X}},$$

with $c_1, c_2 > 0$. While this does not seem particularly helpful at first glance, it in fact allows to drastically change the practical performance of CSG. By altering the ratio $\xi = \frac{c_2}{c_1}$, the CSG gradient approximation tends to consider fewer ($\xi < 1$) or more ($\xi > 1$) old samples in the linear combination. The effect such a choice can have in practice is visible in the numerical analysis of CSG in [27].

Fig. 6 The empirical weights assign each line segment to its empirical measure, i.e., the fraction of points (u_n, x_i) (red stars) which lie inside the corresponding cell

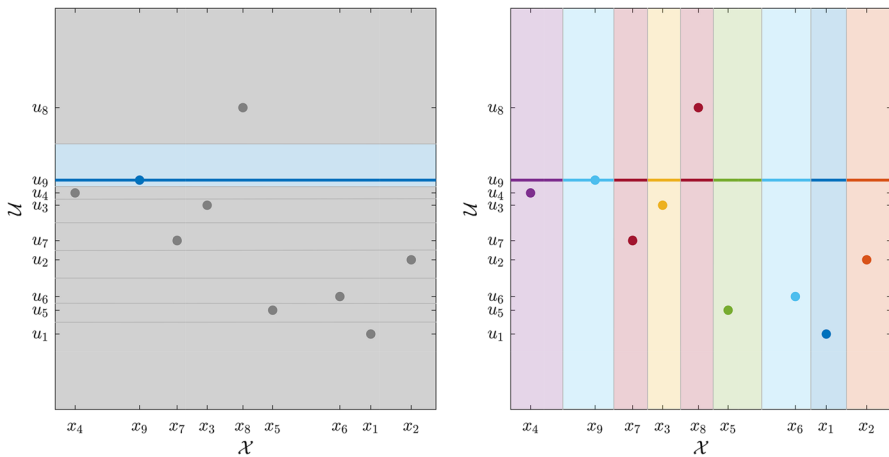
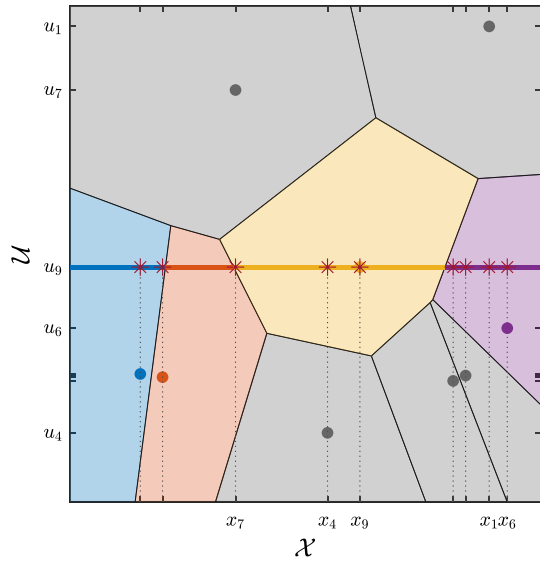


Fig. 7 Voronoi diagrams for the integration weight calculation, given the sample points $(u_i, x_i)_{i=1,\dots,9}$. Different ratios ξ in the norm on $\mathcal{U} \times \mathcal{X}$ lead to SG-like behavior (left, $\xi = 0.01$) or averaging over all samples (right, $\xi = 100$)

To be precise, choosing $\xi \ll 1$ results in the nearest neighbor being predominantly determined by the distance in design. In the extreme case, this means that CSG initially behaves more like a traditional SG algorithm. Analogously, $\xi \gg 1$ will initially yield a gradient approximation, in which all old samples are used, even if they differ greatly in design (for discrete measure μ , this corresponds to SAG). The corresponding Voronoi diagrams are shown in Fig. 7.

For the sake of consistency, all numerical examples will use the Euclidean norm $\|\cdot\|_2$ as inner norms and we fix $\xi = 1$.

3.6 Properties of the integration weights

In this section, we will show that each of the previously presented methods of obtaining integration weights satisfies Assumption 2.4. For this, we first have to identify the corresponding measures μ_n and then need to argue why they converge weakly to the true measure μ .

Starting with the empirical weights α^e , we see that, in the sense of Assumption 2.4, they correspond to the empirical measure (see, e.g., [39])

$$\mu_n^e := \frac{1}{n} \sum_{i=1}^n \delta_{x_i},$$

since

$$\begin{aligned} \hat{G}_n^e &= \sum_{i=1}^n \alpha_i^e \nabla_1 j(u_i, x_i) = \sum_{i=1}^n \frac{1}{n} \sum_{m=1}^n 1_{M_i}(x_m) \nabla_1 j(u_i, x_i) \\ &= \sum_{i=1}^n \frac{1}{n} \sum_{m=1}^n \delta_{k^n(x_m)}(i) \nabla_1 j(u_i, x_i) = \sum_{i=1}^n \int_{\mathcal{X}} \delta_{k^n(x)}(i) \nabla_1 j(u_i, x_i) \mu_n^e(dx). \end{aligned}$$

It was shown in [40, Theorem 3], that $\mu_n^e \Rightarrow \mu$ for $n \rightarrow \infty$, where \Rightarrow denotes the weak convergence of measures (in our case, the weak-* convergence in dual space theory, see, e.g., [41, Section 4.3]):

$$\nu_n \Rightarrow \nu \quad \text{iff} \quad \int_{\mathcal{X}} f(x) \nu_n(dx) \rightarrow \int_{\mathcal{X}} f(x) \nu(dx) \quad \text{for all } f \in \mathcal{C}^{0,1}(\mathcal{X}, \mathbb{R}).$$

Likewise, the measures associated with the other integration weights are given by

$$\mu_n^{\text{ex}} = \mu, \quad \mu_n^{\text{eh}} = \sum_{i=1}^n \delta_{x_i} \mu(\tilde{M}_i) \quad \text{and} \quad \mu_n^{\text{ih}} = \sum_{i=1}^n \delta_{x_{j_i}} \mu_{f(n)}^e(\tilde{M}_{j_i}).$$

Now, $\mu_n^{\text{ex}} \Rightarrow \mu$ is clear. For μ_n^{eh} , given any $f \in \mathcal{C}^{0,1}(\mathcal{X}, \mathbb{R})$, we have

$$\begin{aligned} \left| \int_{\mathcal{X}} f(x) \mu_n^{\text{eh}}(dx) - \int_{\mathcal{X}} f(x) \mu(dx) \right| &= \left| \sum_{i=1}^n f(x_i) \mu(\tilde{M}_i) - \int_{\mathcal{X}} f(x) \mu(dx) \right| \\ &= \left| \sum_{i=1}^n \int_{\tilde{M}_i} (f(x_i) - f(x)) \mu(dx) \right| \\ &\leq \sum_{i=1}^n \int_{\tilde{M}_i} L_f \|x - x_i\|_{\mathcal{X}} \mu(dx) \end{aligned}$$

$$\begin{aligned}
&\leq L_f \sum_{i=1}^n \text{diam}(\tilde{M}_i \cap \text{supp}(\mu)) \int_{\tilde{M}_i} \mu(dx) \\
&\leq L_f \max_{i=1, \dots, n} \{\text{diam}(\tilde{M}_i \cap \text{supp}(\mu))\} \rightarrow 0.
\end{aligned}$$

Here, we used the almost sure density of $(x_n)_{n \in \mathbb{N}}$, see Remark 2.1, together with the definition of \tilde{M}_i , to conclude

$$\text{diam}(\tilde{M}_i \cap \text{supp}(\mu)) := \sup_{\substack{x, y \in \tilde{M}_i \\ x, y \in \text{supp}(\mu)}} \|x - y\|_{\mathcal{X}} \xrightarrow{n \rightarrow \infty} 0 \quad \text{for all } i \in \mathbb{N}.$$

Lastly, $\mu_n^{\text{ih}} \Rightarrow \mu$ follows from combining $\mu_n^{\text{eh}} \Rightarrow \mu$ and $\mu_n^e \Rightarrow \mu$ with the triangle inequality.

3.7 Batches, patches and parallelization

All methods for obtaining the integration weights presented above heavily relied on evaluating the gradient $\nabla_1 j$ only at a single sample point (u_n, x_n) per iteration. In other words, Algorithm 1 has a natural batch size of 1. While we certainly do not want to increase this number too much, stochastic mini-batch algorithms outperform classic SG in some instances, especially if the evaluation of the gradient samples $\nabla_1 j(u_n, x_n^{(i)})_{i=1, \dots, N}$ can be done in parallel.

Increasing the batch size N in CSG leaves the question of how the integration weights should be obtained. We could, of course, simply collect all evaluation points and calculate the weights as usual. This, however, may significantly increase the computational cost, effectively scaling it by N . In many instances, there is a much more elegant solution to this problem, which lets us include mini-batches without any additional weight calculation cost.

Assume, for simplicity, that \mathcal{X} is an open interval and that μ corresponds to a uniform distribution, i.e., there exist $a < b \in \mathbb{R}$ such that

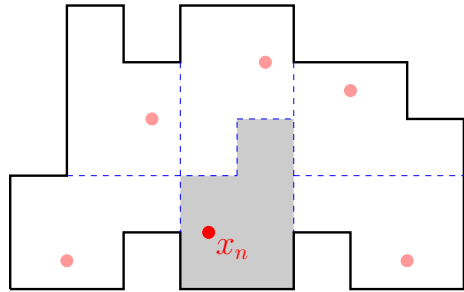
$$J(u) = \frac{1}{b-a} \int_a^b j(u, x) dx.$$

Given $N \in \mathbb{N}$, we obtain

$$\begin{aligned}
J(u) &= \frac{1}{b-a} \int_a^b j(u, x) dx = \frac{1}{b-a} \sum_{i=1}^N \int_{a+(i-1)\frac{b-a}{N}}^{a+i\frac{b-a}{N}} j(u, x) dx \\
&= \frac{1}{b-a} \int_a^{a+\frac{b-a}{N}} \sum_{i=1}^N j(u, x + (i-1)\frac{b-a}{N}) dx =: \frac{1}{b-a} \int_a^{a+\frac{b-a}{N}} \tilde{j}(u, x) dx.
\end{aligned}$$

Thus, we can include mini-batches of size $N \in \mathbb{N}$ into CSG by performing the following steps in each iteration:

Fig. 8 To obtain a mini-batch of six samples points, the domain \mathcal{X} is subdivided into six patches, indicated by the blue lines. In each iteration, a sample point x_n is drawn in patch \mathcal{X}_1 (grey) and afterwards translated into all other patches (Color figure online)



- 1.) Draw random sample $x_n \in (a, a + \frac{b-a}{N})$.
- 2.) Evaluate $\nabla_1 j(u_n, \cdot)$ at each

$$x_n^{(i)} = x_n + (i - 1) \frac{b-a}{N}, \quad i = 1, \dots, N.$$

- 3.) Compute

$$\nabla_1 \tilde{j}(u_n, x_n) = \sum_{i=1}^N \nabla_1 j(u_n, x_n^{(i)}).$$

- 4.) Calculate integration weights $(\alpha_k)_{k=1, \dots, n}$ as usual for $(u_k, x_k)_{k=1, \dots, n}$ and set

$$\hat{G}_n = \sum_{k=1}^n \alpha_k \nabla_1 \tilde{j}(u_k, x_k).$$

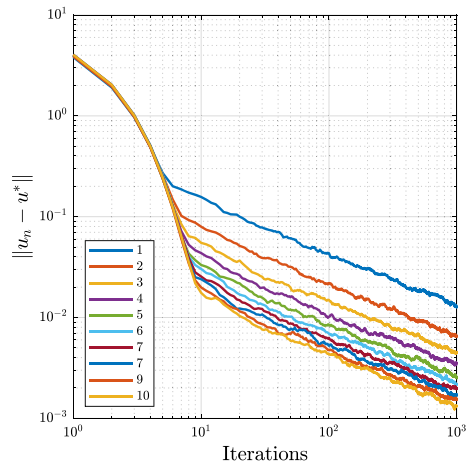
It is straightforward to apply this idea to higher dimensional rectangular cuboids. Furthermore, the process of subdividing \mathcal{X} into smaller patches and drawing the samples in only one of these regions $\mathcal{X}_1 \subseteq \mathcal{X}$ can be generalized to more complex settings as well. However, it is necessary that translating the sample x_n into the other patches preserves the underlying probability distribution $X \sim \mu$, e.g., if μ is induced by a Lipschitz continuous density function and the different patches are obtained by reflecting, translating, rotating and scaling \mathcal{X}_1 . A conceptual example is shown in Fig. 8.

The effect of introducing mini-batches into CSG is tested for the optimization problem

$$\min_{u \in [-5, 5]^2} \frac{1}{2} \int_{\mathcal{X}} \|u - x\|_2^2 dx, \quad (4)$$

by dividing $\mathcal{X} = [-\frac{1}{2}, \frac{1}{2}]^2$ into small squares of sidelength $\frac{1}{N}$, achieving a batch size of N^2 . The results of 500 optimization runs with random starting points are given in Fig. 9. Although increasing the batch size does, as expected, not influence the rate of convergence, it still improves the overall performance and should definitely be considered for complex optimization problems, especially if the gradient sampling can be performed in parallel.

Fig. 9 Median distance of the CSG iterates to the optimal solution $u^* = 0 \in \mathbb{R}^2$ of (4) in each iteration. The batch size used is equal to N^2 , where different values of N are indicated by the different colors. For all runs, a constant step size of $\tau = \frac{1}{2}$ was chosen (Color figure online)



4 Auxiliary results

From now on, unless explicitly stated otherwise, we will always assume Assumptions 2.1 to 2.4 to be satisfied. In this section, we collect some auxiliary results known from the convergence analysis of gradient descent schemes, like Lemma 4.1. Afterwards, we focus on more specific results concerning the density of the sample point sequence $(u_n, x_n)_{n \in \mathbb{N}}$, that are later used to prove the approximation property of CSG (Lemma 4.6). Said property represents the most distinguishing feature between CSG and other stochastic optimization methods and plays the most important role for our later convergence results in Sects. 5 and 6. Note that similar results were already obtained in [1]. Therein, however, the argumentation relied heavily on the usage of diminishing step sizes and exact integration weights. The results presented here hold in a more general setting and allow for the analysis of continuous stochastic optimization schemes beyond pure gradient methods in the future, see Remark 4.2.

Definition 4.1 (*Stationary Points*) Let $J \in C^1(\mathcal{U})$ be given. We say $u^* \in \mathcal{U}$ is a stationary point of J , if

$$\mathcal{P}_{\mathcal{U}}(u^* - t \nabla J(u^*)) = u^* \quad \text{for all } t > 0. \quad (5)$$

Furthermore, we denote by

$$\mathcal{S}(J) := \{u \in \mathcal{U} : \mathcal{P}_{\mathcal{U}}(u - t \nabla J(u)) = u \text{ for all } t > 0\}$$

the set of all stationary points of J . Note that, under Assumption 2.2, (5) is equivalent to

$$\mathcal{P}_{\mathcal{U}}(u^* - t \nabla J(u^*)) = u^* \quad \text{for some } t > 0.$$

Gradient descent methods for L -smooth objective functions have thoroughly been studied in the past (e.g. [42, 43]). The key ingredients for obtaining convergence results

with constant step sizes are the descent lemma and the characteristic property of the projection operator, which we state in the following.

Lemma 4.1 (Descent Lemma) *If $J : \mathcal{U} \rightarrow \mathbb{R}$ is L -smooth, then it holds*

$$J(u_1) \leq J(u_2) + \nabla J(u_2)^\top (u_1 - u_2) + \frac{L}{2} \|u_1 - u_2\|_{\mathcal{U}}^2 \quad \forall u_1, u_2 \in \mathcal{U}.$$

Lemma 4.2 (Characteristic Property of Projection) *For a projected step in direction \hat{G}_n , i.e., $u_{n+1} = \mathcal{P}_{\mathcal{U}}(u_n - \tau_n \hat{G}_n)$, the following holds:*

$$\hat{G}_n^\top (u_n - u_{n+1}) \geq \frac{\|u_n - u_{n+1}\|_{\mathcal{U}}^2}{\tau_n}.$$

Proof Lemma 4.1 corresponds to [44, Lemma 5.7]. Lemma 4.2 is a direct consequence of [44, Theorem 6.41]. \square

Before we move on to results that are specific for the CSG method, we state a general convergence result, which will be helpful in the later proofs.

Lemma 4.3 (Finitely Many Accumulation Points) *Let $(u_n)_{n \in \mathbb{N}} \subseteq \mathbb{R}^d$ be a bounded sequence. Suppose that $(u_n)_{n \in \mathbb{N}}$ has only finitely many accumulation points and it holds $\|u_{n+1} - u_n\| \rightarrow 0$. Then $(u_n)_{n \in \mathbb{N}}$ is convergent.*

Proof Let $\{\bar{u}_1, \dots, \bar{u}_K\}$ be the accumulation points of $(u_n)_{n \in \mathbb{N}}$ and define

$$\delta_0 := \min_{\substack{i, j \in \{1, \dots, K\} \\ i \neq j}} \|\bar{u}_i - \bar{u}_j\|,$$

i.e., the minimal distance between two accumulation points of $(u_n)_{n \in \mathbb{N}}$. The accumulation point closest to u_n is defined as:

$$\bar{u}(n) := \arg \min_{u \in \{\bar{u}_1, \dots, \bar{u}_K\}} \|u_n - u\|.$$

Next up, we show that there exists $N \in \mathbb{N}$ such that for all $n \geq N$ it holds $\|u_n - \bar{u}(n)\| < \frac{\delta_0}{4}$. We prove this by contradiction.

Thus, we assume there exist infinitely many $n \in \mathbb{N}$ such that $\|u_n - \bar{u}(n)\| \geq \frac{\delta_0}{4}$. This subsequence is again bounded and therefore must have an accumulation point. By construction, this accumulation point is no accumulation point of $(u_n)_{n \in \mathbb{N}}$, which is a contradiction.

Now, let $N_1 \in \mathbb{N}$ be large enough such that $\|u_n - \bar{u}(n)\| < \frac{\delta_0}{4}$ for all $n \geq N_1$. By our assumptions, there also exists $N_2 \in \mathbb{N}$ with $\|u_{n+1} - u_n\| < \frac{\delta_0}{4}$ for all $n \geq N_2$. Define $N := \max\{N_1, N_2\}$.

Let $n \geq N$ and assume for contradiction that $\bar{u}(n) \neq \bar{u}(n+1)$. We obtain

$$\text{dist}(\mathcal{B}_{\delta_0/4}(\bar{u}(n)), \mathcal{B}_{\delta_0/4}(\bar{u}(n+1))) \geq \frac{\delta_0}{2} > \frac{\delta_0}{4} > \|u_n - u_{n+1}\| \quad \text{for all } n \geq N,$$

where $\text{dist}(A, B) := \inf_{x \in A, y \in B} \|x - y\|$ for $A, B \subseteq \mathbb{R}^d$. This is a contradiction to $\|u_{n+1} - u_n\| < \frac{\delta_0}{4}$ for all $n \geq N$.

We thus conclude that $u_n \in \mathcal{B}_{\delta_0/4}(\bar{u}(n))$ implies $u_{n+1} \in \mathcal{B}_{\delta_0/4}(\bar{u}(n))$ as well. Since $\bar{u}(n)$ is the only accumulation point on $\mathcal{B}_{\delta_0/4}(\bar{u}(n))$, it follows that $u_n \rightarrow \bar{u}(N)$. \square

4.1 Results for CSG approximations

From now on, let $(u_n)_{n \in \mathbb{N}}$ denote the sequence of iterates generated by Algorithm 1. In this section, we want to show that the CSG approximations \hat{J}_n and \hat{G}_n in the course of iterations approach the values of $J(u_n)$ and $\nabla J(u_n)$, respectively. This is a key result for the convergence theorems stated in Sect. 5 and Sect. 6.

Lemma 4.4 (Density Result in \mathcal{X}) *Let $(x_n)_{n \in \mathbb{N}}$ be the random sequence appearing in Algorithm 1. For all $\varepsilon > 0$ there exists $N \in \mathbb{N}$ such that*

$$\min_{n \in \{1, \dots, N\}} \|x_n - x\|_{\mathcal{X}} < \varepsilon \text{ for all } x \in \text{supp}(\mu).$$

Proof Utilizing the compactness of $\text{supp}(\mu) \subseteq \mathbb{R}^{d_r}$, there exists $T \in \mathbb{N}$ such that $(\mathcal{B}_{\varepsilon/2}(m_i))_{i=1, \dots, T}$ is an open cover of $\text{supp}(\mu)$ consisting of balls with radius $\frac{\varepsilon}{2}$ centered at points $m_i \in \text{supp}(\mu)$. Thus, for each $x \in \text{supp}(\mu)$ we can find $i_x \in \{1, \dots, T\}$ with $x \in \mathcal{B}_{\varepsilon/2}(m_{i_x})$. Hence, by Remark 2.1, for each $i = 1, \dots, T$, there exists $n_i \in \mathbb{N}$ satisfying

$$\|x_{n_i} - m_i\|_{\mathcal{X}} < \frac{\varepsilon}{2}.$$

Defining

$$N := \max_{i \in \{1, \dots, T\}} n_i < \infty,$$

for all $x \in \text{supp}(\mu)$ we have

$$\begin{aligned} \min_{n \in \{1, \dots, N\}} \|x - x_n\|_{\mathcal{X}} &\leq \min_{n \in \{1, \dots, N\}} (\|x - m_{i_x}\|_{\mathcal{X}} + \|m_{i_x} - x_n\|_{\mathcal{X}}) \\ &< \frac{\varepsilon}{2} + \min_{n \in \{1, \dots, N\}} \|m_{i_x} - x_n\|_{\mathcal{X}} < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned}$$

\square

Lemma 4.5 (Density Result in $\mathcal{U} \times \mathcal{X}$) *Let $(u_n)_{n \in \mathbb{N}}$, $(x_n)_{n \in \mathbb{N}}$ be the sequences of optimization variables and sample sequence appearing in Algorithm 1. For all $\varepsilon > 0$ there exists $N \in \mathbb{N}$ such that*

$$Z_n(x) := \min_{k \in \{1, \dots, n\}} (\|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}}) < \varepsilon$$

for all $n > N$ and all $x \in \text{supp}(\mu)$.

Proof Since \mathcal{U} is compact, we can find a finite cover $(\mathcal{B}_{\varepsilon/4}(m_i))_{i=1,\dots,T}$ of \mathcal{U} consisting of $T \in \mathbb{N}$ balls with radius $\frac{\varepsilon}{4}$ centered at points $m_i \in \mathcal{U}$. Define $I \subseteq \{1, \dots, T\}$ as

$$I := \{i \in \{1, \dots, T\} : u_n \in \mathcal{B}_{\varepsilon/4}(m_i) \text{ for infinitely many } n \in \mathbb{N}\}.$$

By our definition of I , for each $i \in \{1, \dots, T\} \setminus I$ there exists $\tilde{N}_i \in \mathbb{N}$ such that $u_n \notin \mathcal{B}_{\varepsilon/4}(m_i)$ for all $n > \tilde{N}_i$. Setting

$$N_1 := \max_{i \in \{1, \dots, T\} \setminus I} \tilde{N}_i,$$

it follows that

$$u_n \notin \mathcal{B}_{\varepsilon/4}(m_i) \text{ for each } n > N_1 \text{ and all } i \in \{1, \dots, T\} \setminus I. \quad (6)$$

For $i \in I$ let $(u_{n_t^{(i)}})_{t \in \mathbb{N}}$ be the subsequence consisting of all elements of $(u_n)_{n \in \mathbb{N}}$ that lie in $\mathcal{B}_{\varepsilon/4}(m_i)$. Observe that $(x_{n_t^{(i)}})_{t \in \mathbb{N}}$ is independent and identically distributed according to μ , since for all $A \subseteq \text{supp}(\mu)$ and each $i \in I$, it holds

$$\mathbb{P}(x_n \in A \mid u_n \in \mathcal{B}_{\varepsilon/4}(m_i)) = \mu(A) \quad \text{for all } n \in \mathbb{N}.$$

Thus, by Remark 2.1, $(x_{n_t^{(i)}})_{t \in \mathbb{N}}$ is dense in $\text{supp}(\mu)$ with probability 1 for all $i \in I$. By Lemma 4.4, we can find $K_i \in \mathbb{N}$ such that

$$\min_{t \in \{1, \dots, K_i\}} \|x - x_{n_t^{(i)}}\|_{\mathcal{X}} < \frac{\varepsilon}{2} \quad \text{for all } x \in \text{supp}(\mu). \quad (7)$$

Define

$$N_2 := \max_{i \in I} \max_{t \in \{1, \dots, K_i\}} n_t^{(i)}$$

as well as $N := \max(N_1, N_2)$. Notice that this definition of N implies for all $i \in I$ and all $n > N$

$$\{n_t^{(i)} : t \in \{1, \dots, K_i\}\} \subseteq \{1, \dots, n\}.$$

By (6), for all $n > N$ there exists $i \in I$ such that

$$\|u_n - m_i\|_{\mathcal{U}} < \frac{\varepsilon}{4}.$$

Now, given $x \in \text{supp}(\mu)$ and $n > N$, we choose $j \in I$ such that $u_n \in \mathcal{B}_{\varepsilon/4}(m_j)$. Thus, it holds

$$\begin{aligned} Z_n(x) &= \min_{k \in \{1, \dots, n\}} (\|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}}) \\ &\leq \min_{t \in \{1, \dots, K_j\}} (\|u_n - u_{n_t^{(j)}}\|_{\mathcal{U}} + \|x - x_{n_t^{(j)}}\|_{\mathcal{X}}) \\ &\leq \min_{t \in \{1, \dots, K_j\}} (\|u_n - m_j\|_{\mathcal{U}} + \|m_j - u_{n_t^{(j)}}\|_{\mathcal{U}} + \|x - x_{n_t^{(j)}}\|_{\mathcal{X}}) \\ &< \frac{\varepsilon}{4} + \frac{\varepsilon}{4} + \frac{\varepsilon}{2} = \varepsilon, \end{aligned}$$

where we used (7) and $u_n, u_{n_t^{(j)}} \in \mathcal{B}_{\varepsilon/4}(m_j)$ in the last line. \square

Lemma 4.6 (Approximation Results for J and ∇J) *The approximation errors for ∇J and J vanish in the course of the iterations, i.e.,*

$$\|\hat{G}_n - \nabla J(u_n)\| \rightarrow 0 \quad \text{and} \quad |\hat{J}_n - J(u_n)| \rightarrow 0 \quad \text{for } n \rightarrow \infty.$$

Proof Denote by v_n the measure corresponding to the integration weights according to (3). For $x \in \text{supp}(\mu)$, we define the closest index in the current iteration $k^n(x)$ as

$$k^n(x) := \arg \min_{k=1, \dots, n} (\|u_n - u_k\|_{\mathcal{U}} + \|x - x_k\|_{\mathcal{X}}),$$

i.e., we have

$$\|u_n - u_{k^n(x)}\|_{\mathcal{U}} + \|x - x_{k^n(x)}\|_{\mathcal{X}} = Z_n(x).$$

Now, it holds

$$\begin{aligned} &\|\hat{G}_n - \nabla J(u_n)\| \\ &= \left\| \sum_{i=1}^n \int_{\mathcal{X}} \delta_{k^n(x)}(i) \nabla_1 j(u_i, x_i) v_n(dx) - \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx) \right\| \\ &\leq \left\| \int_{\mathcal{X}} \left(\sum_{i=1}^n \delta_{k^n(x)}(i) \nabla_1 j(u_i, x_i) - \nabla_1 j(u_n, x) \right) v_n(dx) \right\| \\ &\quad + \left\| \int_{\mathcal{X}} \nabla_1 j(u_n, x) v_n(dx) - \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx) \right\| \\ &\leq L_j \int_{\mathcal{X}} Z_n(x) v_n(dx) + \left\| \int_{\mathcal{X}} \nabla_1 j(u_n, x) v_n(dx) - \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx) \right\|, \end{aligned}$$

where L_j is the Lipschitz constant of $\nabla_1 j$ as defined in Assumption 2.3.

First, since Z_n is uniformly (in n) Lipschitz continuous, we obtain

$$\begin{aligned} \int_{\mathcal{X}} Z_n(x) v_n(dx) &= \int_{\mathcal{X}} Z_n(x) \mu(dx) + \int_{\mathcal{X}} Z_n(x) v_n(dx) - \int_{\mathcal{X}} Z_n(x) \mu(dx) \\ &\leq \int_{\mathcal{X}} Z_n(x) \mu(dx) + L_Z \cdot d_W(v_n, \mu). \end{aligned}$$

Here, L_Z corresponds to the Lipschitz constant of Z_n and d_W denotes the Wasserstein distance of the measure v_n and μ . By Assumption 2.2, \mathcal{X} is bounded and by Assumption 2.4, we have $v_n \Rightarrow \mu$. Thus, [45, Theorem 6] yields $d_W(v_n, \mu) \rightarrow 0$. Additionally, since Z_n is bounded and converges pointwise to 0 (see Lemma 4.5), we use Lebesgue's dominated convergence theorem and conclude

$$\int_{\mathcal{X}} Z_n(x) \mu(dx) \rightarrow 0 \quad \text{for } n \rightarrow \infty.$$

For the second part, let Q_n be an arbitrary coupling of v_n and μ , i.e., $Q_n(\cdot \times \mathcal{X}) = v_n$ and $Q_n(\mathcal{X} \times \cdot) = \mu$. Utilizing the Lipschitz continuity of $\nabla_1 j$ (Assumption 2.3) once again, we obtain

$$\begin{aligned} &\left\| \int_{\mathcal{X}} \nabla_1 j(u_n, x) v_n(dx) - \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx) \right\| \\ &\leq \left\| \int_{\mathcal{X} \times \mathcal{X}} (\nabla_1 j(u_n, x) - \nabla_1 j(u_n, y)) Q_n(d(x, y)) \right\| \\ &\leq L_j \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_{\mathcal{X}} Q_n(d(x, y)). \end{aligned}$$

Denote the set of all couplings of v_n and μ by \mathbf{Q} . Since Q_n was arbitrary, it holds

$$\begin{aligned} &\left\| \int_{\mathcal{X}} \nabla_1 j(u_n, x) v_n(dx) - \int_{\mathcal{X}} \nabla_1 j(u_n, x) \mu(dx) \right\| \\ &\leq L_j \cdot \inf_{Q_n \in \mathbf{Q}} \int_{\mathcal{X} \times \mathcal{X}} \|x - y\|_{\mathcal{X}} Q_n(d(x, y)) \\ &= L_j \cdot d_W(v_n, \mu) \rightarrow 0 \quad \text{for } n \rightarrow \infty, \end{aligned}$$

finishing the proof of $\|\hat{G}_n - \nabla J(u_n)\| \rightarrow 0$. The second part of the claim follows analogously. \square

As a final remark before starting the convergence analysis, we want to give further details on the class of problems that can be solved by the CSG algorithm.

Remark 4.1 (*Generalized Setting*) Suppose that, in addition to \mathcal{U} , \mathcal{X} and J as defined in the introduction, we are given a convex set $\mathcal{V} \subseteq \mathbb{R}^{d_1}$ for some $d_1 \in \mathbb{N}$ and a continuously differentiable function $F : \mathcal{V} \times \mathbb{R} \rightarrow \mathbb{R}$. Now, if we consider the optimization problem

$$\min_{(u,v) \in \mathcal{U} \times \mathcal{V}} F(v, J(u)), \quad (8)$$

the gradient of the objective function with respect to (u, v) is given by

$$\nabla F(v, J(u)) = \begin{pmatrix} \nabla_1 F(v, J(u)) \\ \nabla J(u) \partial_2 F(v, J(u)) \end{pmatrix}.$$

It is a direct consequence of Lemma 4.6, that

$$\left\| \begin{pmatrix} \nabla_1 F(v_n, \hat{J}_n) \\ \hat{G}_n \partial_2 F(v_n, \hat{J}_n) \end{pmatrix} - \nabla F(v_n, J(u_n)) \right\| \rightarrow 0 \quad \text{for } n \rightarrow \infty.$$

Thus, we can use the CSG method to solve (8) and all our convergence results carry over to this setting, as long as the new objective function satisfies Assumption 2.3.

Furthermore, let $\mathcal{Y} \subseteq \mathbb{R}^{d_2}$ for some $d_2 \in \mathbb{N}$. Assume that we are given a probability measure ν such that the pair (\mathcal{Y}, ν) satisfies the same assumptions we imposed on (\mathcal{X}, μ) and consider the optimization problem

$$\min_{(u,v) \in \mathcal{U} \times \mathcal{V}} \int_{\mathcal{Y}} \tilde{F}(v, J(u), y) \nu(dy). \quad (9)$$

Again, the gradient of this objective function can be approximated by the CSG method, if $\tilde{F} : \mathcal{V} \times \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$ is Lipschitz continuously differentiable.

It is clear that we can continue to wrap around functions or expectation values in these fashions indefinitely. Therefore, we see that the scope of the CSG method is far larger than problems like (1) and includes many settings, which stochastic gradient descent methods can not handle, like nested expected values, tracking of expected values and many more.

A numerical example for a composite objective function, where CSG is compared to a method from the literature, can be found in Sect. 7.2.

Remark 4.2 Carefully observing the proofs above, it can be seen that the approximation property (Lemma 4.6) almost surely holds for any sequence of designs, regardless of the method it is obtained by. Thus, the results presented in this section are a consequence of the underlying stochastic approximation scheme through our integration weights and do not depend on the outer gradient descent method, in which they are used. Therefore, for future work, we may even consider combining the approximations \hat{J}_n and \hat{G}_n with more advanced optimization schemes from literature.

5 Convergence results for constant step size

Our first result considers the special case in which the objective function J appearing in (1) has only finitely many stationary points on \mathcal{U} . The proof of this result serves as a prototype for the later convergence results, as they share a common idea.

To illustrate the practical performance of CSG with constant step sizes, an academic example is presented in Sect. 7.1.

Theorem 5.1 (Convergence for Constant Steps) *Assume that J has only finitely many stationary points on \mathcal{U} .*

Then CSG with a positive constant step size $\tau_n = \tau < \frac{2}{L}$ converges to a stationary point of J with probability 1.

We want to sketch the proof of Theorem 5.1 before going into details. In the deterministic case, Lemma 4.1 and Lemma 4.2 are used to show that $J(u_{n+1}) \leq J(u_n)$ for all $n \in \mathbb{N}$. It then follows from a telescopic sum argument that $\|u_{n+1} - u_n\| \rightarrow 0$, i.e., every accumulation point of $(u_n)_{n \in \mathbb{N}}$ is stationary (compare [43, Theorem 5.1] or [44, Theorem 10.15]).

In the case of CSG, we can not guarantee monotonicity of the objective function values $(J(u_n))_{n \in \mathbb{N}}$. Instead, we split the sequence into two subsequences. On one of these subsequences, we can guarantee a decrease in function values, while for the other sequence we can not. However, we prove that the latter sequence can accumulate only at stationary points of J . The main idea is then that $(u_n)_{n \in \mathbb{N}}$ can have only one accumulation point, because “switching” between several points conflicts with the decrease in function values that must happen for steps in between.

Proof of Theorem 5.1 The following arguments are being made for a fixed sample sequence $(x_n)_{n \in \mathbb{N}}$ and the corresponding trajectory of designs $(u_n)_{n \in \mathbb{N}}$. Moreover, during the proof, we construct several subsequences of $(u_n)_{n \in \mathbb{N}}$. By construction and Assumption 2.1, the associated subsequences of $(x_n)_{n \in \mathbb{N}}$ are again independent identically distributed realizations of $X \sim \mu$. Thus, each of these subsequences is dense in $\text{supp}(\mu)$ with probability 1. Since this property is required for the following arguments, we assume that $(x_n)_{n \in \mathbb{N}}$ satisfies this condition.

By Lemma 4.1 we have

$$\begin{aligned} J(u_{n+1}) - J(u_n) &\leq \nabla J(u_n)^\top (u_{n+1} - u_n) + \frac{L}{2} \|u_{n+1} - u_n\|_{\mathcal{U}}^2 \\ &= \hat{G}_n^\top (u_{n+1} - u_n) + \frac{L}{2} \|u_{n+1} - u_n\|_{\mathcal{U}}^2 + \left(\nabla J(u_n) - \hat{G}_n \right)^\top (u_{n+1} - u_n). \end{aligned}$$

Utilizing Lemma 4.2 and the Cauchy-Schwarz inequality, we now obtain

$$\begin{aligned} J(u_{n+1}) - J(u_n) &\leq \left(\frac{L}{2} - \frac{1}{\tau} \right) \|u_{n+1} - u_n\|_{\mathcal{U}}^2 + \left\| \nabla J(u_n) - \hat{G}_n \right\| \cdot \|u_{n+1} - u_n\|_{\mathcal{U}} \\ &= \left(\left(\frac{L}{2} - \frac{1}{\tau} \right) \|u_{n+1} - u_n\|_{\mathcal{U}} + \left\| \nabla J(u_n) - \hat{G}_n \right\| \right) \|u_{n+1} - u_n\|_{\mathcal{U}}. \quad (10) \end{aligned}$$

Since $\frac{L}{2} - \frac{1}{\tau} < 0$, our idea is the following:

Steps that satisfy

$$\left\| \nabla J(u_n) - \hat{G}_n \right\| \leq \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{n+1} - u_n\|_{\mathcal{U}}, \quad (11)$$

i.e., steps with small errors in the gradient approximation, will yield decreasing function values.

On the other hand, the remaining steps will satisfy $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$, due to $\|\nabla J(u_n) - \hat{G}_n\| \rightarrow 0$ (see Lemma 4.6). With this in mind, we distinguish three cases: In Case 1, (11) is satisfied for almost all steps, while in Case 2 it is satisfied for only finitely many steps. In the last case, there are infinitely many steps satisfying and infinitely many steps violating (11).

Case 1: There exists $N \in \mathbb{N}$ such that

$$\|\nabla J(u_n) - \hat{G}_n\| \leq \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{n+1} - u_n\|_{\mathcal{U}} \quad \text{for all } n \geq N.$$

In this case, it follows from (10) that $J(u_{n+1}) \leq J(u_n)$ for all $n \geq N$. Therefore, the sequence $(J(u_n))_{n \in \mathbb{N}}$ is monotonically decreasing for almost every $n \in \mathbb{N}$. Since J is continuous and \mathcal{U} is compact, J is bounded and we therefore have $J(u_n) \rightarrow \bar{J} \in \mathbb{R}$. Thus, it holds

$$-\infty < \bar{J} - J(u_N) = \sum_{n=N}^{\infty} (J(u_{n+1}) - J(u_n)) \leq \frac{1}{2} \left(\frac{L}{2} - \frac{1}{\tau} \right) \sum_{n=N}^{\infty} \|u_{n+1} - u_n\|_{\mathcal{U}}^2.$$

Since $\frac{1}{2} \left(\frac{L}{2} - \frac{1}{\tau} \right) < 0$, we must have $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$. Let $(u_{n_k})_{k \in \mathbb{N}}$ be a convergent subsequence with $u_{n_k} \rightarrow \bar{u} \in \mathcal{U}$.

By Lemma 4.6, we have $\hat{G}_{n_k} \rightarrow \nabla J(\bar{u})$ and thus

$$\begin{aligned} 0 &= \lim_{k \rightarrow \infty} \|u_{n_k+1} - u_{n_k}\|_{\mathcal{U}} \\ &= \lim_{k \rightarrow \infty} \|\mathcal{P}_{\mathcal{U}}(u_{n_k} - \tau \hat{G}_{n_k}) - u_{n_k}\|_{\mathcal{U}} \\ &= \|\mathcal{P}_{\mathcal{U}}(\bar{u} - \tau \nabla J(\bar{u})) - \bar{u}\|_{\mathcal{U}}, \end{aligned}$$

i.e., every accumulation point of $(u_n)_{n \in \mathbb{N}}$ is stationary. Since J has only finitely many stationary points, Lemma 4.3 yields the convergence of $(u_n)_{n \in \mathbb{N}}$ to a stationary point of J .

Case 2: There exists $N \in \mathbb{N}$ such that

$$\|\nabla J(u_n) - \hat{G}_n\| > \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{n+1} - u_n\|_{\mathcal{U}} \quad \text{for all } n \geq N.$$

By Lemma 4.6, we have $\|\nabla J(u_n) - \hat{G}_n\| \rightarrow 0$. Since $\frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) > 0$, the above inequality directly implies $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$. Analogously to Case 1, we conclude that $(u_n)_{n \in \mathbb{N}}$ converges to a stationary point of J .

Case 3: There are infinitely many $n \in \mathbb{N}$ with

$$\|\nabla J(u_n) - \hat{G}_n\| \leq \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{n+1} - u_n\|_{\mathcal{U}}$$

and infinitely many $n \in \mathbb{N}$ with

$$\left\| \nabla J(u_n) - \hat{G}_n \right\| > \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{n+1} - u_n\|_{\mathcal{U}}.$$

In this case, we split $(u_n)_{n \in \mathbb{N}}$ disjointly in the two sequences $(u_{a(n)})_{n \in \mathbb{N}}$ and $(u_{b(n)})_{n \in \mathbb{N}}$, such that we have

$$\left\| \nabla J(u_{a(n)}) - \hat{G}_{a(n)} \right\| \leq \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{a(n)+1} - u_{a(n)}\|_{\mathcal{U}} \quad \text{for all } n \in \mathbb{N}$$

and

$$\left\| \nabla J(u_{b(n)}) - \hat{G}_{b(n)} \right\| > \frac{1}{2} \left(\frac{1}{\tau} - \frac{L}{2} \right) \|u_{b(n)+1} - u_{b(n)}\|_{\mathcal{U}} \quad \text{for all } n \in \mathbb{N}.$$

We call $(u_{a(n)})_{n \in \mathbb{N}}$ the sequence of descent steps. For $(u_{b(n)})_{n \in \mathbb{N}}$, observe that, as in Case 2, we directly obtain

$$\|u_{b(n)+1} - u_{b(n)}\|_{\mathcal{U}} \rightarrow 0 \quad (12)$$

and every accumulation point of $(u_{b(n)})_{n \in \mathbb{N}}$ is stationary. Therefore, as in the proof of Lemma 4.3, for all $\varepsilon > 0$ there exists $N \in \mathbb{N}$ such that

$$\min_{i \in \{1, \dots, K\}} \|u_{b(n)} - \bar{u}_i\|_{\mathcal{U}} < \varepsilon \quad \text{for all } n \geq N, \quad (13)$$

where $\bar{u}_1, \dots, \bar{u}_K$ denote the $K \in \mathbb{N}$ accumulation points of $(u_{b(n)})_{n \in \mathbb{N}}$.

Now, we prove by contradiction that $J(\bar{u}_1) = J(\bar{u}_2) = \dots = J(\bar{u}_K)$.

Suppose that this is not the case. Then we have at least $M \geq 2$ function values of accumulation points and

$$F := \{J(u) : u = \bar{u}_1, \dots, \bar{u}_K\} = \{f_1, f_2, \dots, f_M\}$$

for some $f_1 > f_2 > \dots > f_M \in \mathbb{R}$. Now, choose $\varepsilon > 0$ small enough, such that

$$2\varepsilon < \min_{\substack{i, j \in \{1, \dots, K\} \\ i \neq j}} \|\bar{u}_i - \bar{u}_j\| \quad \text{and} \quad c_L \varepsilon < f_1 - f_2,$$

where c_L denotes the Lipschitz constant of J . By (12) and (13), there exists $N \in \mathbb{N}$ such that for all $n \geq N$ we have

$$\|u_{b(n)+1} - u_{b(n)}\|_{\mathcal{U}} < \frac{\varepsilon}{4} \quad \text{and} \quad \min_{i \in \{1, \dots, K\}} \|u_{b(n)} - \bar{u}_i\|_{\mathcal{U}} < \frac{\varepsilon}{4}. \quad (14)$$

Therefore, for $n \geq N$ and $i \in \{1, \dots, K\}$, we have

$$u_{b(n)} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i) \implies u_{b(n)+1} \in \mathcal{B}_{\frac{\varepsilon}{2}}(\bar{u}_i) \quad (15)$$

$$\implies u_{b(n)+1} \notin \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_j) \quad \text{for all } j \neq i. \quad (16)$$

Especially, for all $n \geq N$ and all $i = 1, \dots, K$ it holds

$$u_{b(n)} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i) \implies J(u_{b(n)+1}) \leq J(\bar{u}_i) + \frac{c_L \varepsilon}{2}. \quad (17)$$

It follows from (14) and (16) that for $n \geq N + 1$:

- (A) If $u_{b(n)} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$ and $u_{b(n)+1} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_j)$ for some $j \neq i$, then there must be at least one descent step between $u_{b(n)}$ and $u_{b(n)+1}$.
- (B) If $u_{b(n)} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$ and $u_{b(n)-1} \notin \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$, then $u_{b(n)-1}$ must be a descent step.

Observe that (A) follows directly from (14) and (16), as moving from the vicinity of \bar{u}_i to a neighborhood of \bar{u}_j requires that there is an intermediate step u_n with $\min_{i \in \{1, \dots, K\}} \|u_{b(n)} - \bar{u}_i\|_{\mathcal{U}} \geq \frac{\varepsilon}{4}$. Similarly, (B) is just the second condition in (16) reformulated.

Note that, given \bar{u}_j, \bar{u}_k with $J(\bar{u}_j) = f_1$ and $J(\bar{u}_i) \leq f_2$, we have

$$J(u) \leq f_2 + \frac{L_J \varepsilon}{4} < f_1 - \frac{L_J \varepsilon}{4} \leq J(v) \quad \text{for all } u \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i) \text{ and } v \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_j).$$

Thus, starting at $u \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$, we can not reach $v \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_j)$ by descent steps alone. Now, let $i \in \{1, \dots, K\}$ be chosen such that $J(\bar{u}_i) \leq f_2$ and let $n_0 \geq N$ be chosen such that $u_{b(n_0)} \in \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$ and $u_{b(n_0)+1} \notin \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i)$. Using (15) and (17), we obtain

$$\begin{aligned} J(u_{b(n_0)+1}) &\leq J(\bar{u}_i) + \frac{c_L \varepsilon}{2} \\ &\leq f_2 + \frac{c_L \varepsilon}{2} \\ &< f_1 - \frac{c_L \varepsilon}{2} < J(u) \quad \text{for all } u \in \mathcal{B}_{\frac{\varepsilon}{4}}\left(J^{-1}(\{f_1\}) \cap \{\bar{u}_1, \dots, \bar{u}_K\}\right). \end{aligned}$$

Therefore, descent steps can never reach $\mathcal{B}_{\frac{\varepsilon}{4}}\left(J^{-1}(\{f_1\}) \cap \{\bar{u}_1, \dots, \bar{u}_K\}\right)$ again! It follows from item (B), that $u_n \notin \mathcal{B}_{\frac{\varepsilon}{4}}\left(J^{-1}(\{f_1\}) \cap \{\bar{u}_1, \dots, \bar{u}_K\}\right)$ for all $n \geq b(n_0) + 1$, in contradiction to $J^{-1}(\{f_1\}) \cap \{\bar{u}_1, \dots, \bar{u}_K\}$ consisting of at least one accumulation point of $(u_n)_{n \in \mathbb{N}}$. Hence, we have

$$J(\bar{u}_1) = \dots = J(\bar{u}_K) =: \bar{J}. \quad (18)$$

Next, we show that every accumulation point of $(u_{a(n)})_{n \in \mathbb{N}}$ is stationary. We prove this by contradiction.

Assume there exists a non-stationary accumulation point \bar{u} of $(u_{a(n)})_{n \in \mathbb{N}}$. Observe that

$$\min_{i \in \{1, \dots, K\}} \|\bar{u} - \bar{u}_i\|_{\mathcal{U}} > 0.$$

Case 3.1: $J(\bar{u}) < \bar{J}$.

Then, by the same arguments as above, there exists $N \in \mathbb{N}$ and $\varepsilon > 0$ s.t.

$$u_n \notin \bigcup_{i=1}^K \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u}_i) \quad \text{for all } n \geq N.$$

This is a contradiction to $\bar{u}_1, \dots, \bar{u}_K$ being accumulation points of $(u_n)_{n \in \mathbb{N}}$.

Case 3.2: $J(\bar{u}) > \bar{J}$.

In this case, there exists $N \in \mathbb{N}$ and $\varepsilon > 0$ such that $u_n \notin \mathcal{B}_{\frac{\varepsilon}{4}}(\bar{u})$ for all $n \geq N$. This is a contradiction to \bar{u} being an accumulation point of $(u_n)_{n \in \mathbb{N}}$.

Case 3.3: $J(\bar{u}) = \bar{J}$.

Since \bar{u} is an accumulation point of $(u_{a(n)})_{n \in \mathbb{N}}$, there exists a subsequence $(u_{a(n_k)})_{k \in \mathbb{N}}$ with $u_{a(n_k)} \rightarrow \bar{u}$. The sequence $(u_{a(n_k)-1})_{k \in \mathbb{N}}$ is bounded and therefore has at least one accumulation point \bar{u}_{-1} and a subsequence $(u_{a(n_{k_l})-1})_{l \in \mathbb{N}}$ with $u_{a(n_{k_l})-1} \rightarrow \bar{u}_{-1}$. It follows that

$$\begin{aligned} \mathcal{P}_{\mathcal{U}}(\bar{u}_{-1} - \tau \nabla J(\bar{u}_{-1})) &= \lim_{t \rightarrow \infty} \mathcal{P}_{\mathcal{U}}(u_{a(n_{k_l})-1} - \tau \hat{G}_{a(n_{k_l})-1}) \\ &= \lim_{t \rightarrow \infty} u_{a(n_{k_l})} \\ &= \bar{u}. \end{aligned}$$

As \bar{u} is not stationary by our assumption, $\bar{u}_{-1} \neq \bar{u}$ and \bar{u}_{-1} is no stationary point of J . Thus, Lemma 4.1 combined with Lemma 4.2 yields

$$J(\bar{u}) - J(\bar{u}_{-1}) \leq \left(\frac{L}{2} - \frac{1}{\tau}\right) \|\bar{u}_{-1} - \bar{u}\|_{\mathcal{U}}^2 < 0.$$

Therefore, \bar{u}_{-1} is an accumulation point of $(u_{a(n)})_{n \in \mathbb{N}}$, which satisfies $J(\bar{u}_{-1}) > J(\bar{u}) = \bar{J}$. This, however, is impossible, as seen in Case 3.2.

In conclusion, in Case 3, all accumulation points of $(u_n)_{n \in \mathbb{N}}$ are stationary. Thus, on every convergent subsequence we have $\|u_{n_k+1} - u_{n_k}\|_{\mathcal{U}} \rightarrow 0$. Since $(u_n)_{n \in \mathbb{N}}$ is bounded, this already implies $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$. Now, Lemma 4.3 yields the claimed convergence of $(u_n)_{n \in \mathbb{N}}$ to a stationary point of J . \square

The idea of the proof above still applies in the case that J is constant on some parts of \mathcal{U} , i.e., J can have infinitely many stationary points. We obtain the following convergence result:

Theorem 5.2 *Let $\mathcal{S}(J)$ be the set of stationary points of J on U as defined in Definition 4.1. Assume that the set*

$$\mathcal{N} := \{J(u) : u \in \mathcal{S}(J)\} \subseteq \mathbb{R}$$

is of Lebesgue-measure zero. Then, with probability 1, every accumulation point of the sequence generated by CSG with constant step size $\tau < \frac{2}{L}$ is stationary and we have convergence in function values.

Remark 5.1 Comparing Theorem 5.1 and Theorem 5.2, observe that under the weaker assumptions on the set of stationary points of J , we no longer obtain convergence for the whole sequence of iterates. To illustrate why that is the case, consider the function $J : \mathbb{R}^{d_0} \rightarrow \mathbb{R}$ given by $J(u) = \cos(\pi \|u\|_2^2)$ and $\mathcal{U} = \{u \in \mathbb{R}^{d_0} : \|u\|_2^2 \leq \frac{3}{2}\}$. Then, $\mathcal{S}(J) = \{0\} \cup \{u \in \mathcal{U} : \|u\|_2 = 1\}$, i.e., every point on the unit sphere is stationary. Thus, we can not use Lemma 4.3 at the end of the proof to obtain convergence of $(u_n)_{n \in \mathbb{N}}$. Theoretically, it might happen that the iterates $(u_n)_{n \in \mathbb{N}}$ cycle around the unit sphere, producing infinitely many accumulation points, all of which have the same objective function value. This, however, did not occur when testing this example numerically.

Remark 5.2 While the assumption in Theorem 5.2 seems unhandy at first, there is actually a rich theory concerning such properties. For example, Sard's Theorem [46] and generalizations [47] give that the assumption holds if $J \in C^{d_0}$ and \mathcal{U} has smooth boundary. Even though it can be shown that there exist functions, which do not satisfy the assumption (e.g. [48, 49]), such counter-examples need to be precisely constructed and will most likely not appear in any application.

Proof of Theorem 5.2 As in the proof of Theorem 5.1, we consider a fixed sample sequence $(x_n)_{n \in \mathbb{N}}$ and the corresponding trajectory of designs $(u_n)_{n \in \mathbb{N}}$. Again, with probability 1, we can assume all relevant subsequences of $(x_n)_{n \in \mathbb{N}}$ to be dense in $\text{supp}(\mu)$. Now, proceeding analogously as in the proof of Theorem 5.1, we only have to adapt two intermediate results in Case 3:

- (R1) The objective function values of all accumulation points of $(u_{b(n)})_{n \in \mathbb{N}}$ are equal.
- (R2) Every accumulation point of $(u_{a(n)})_{n \in \mathbb{N}}$ is stationary.

Assume first, that (R1) does not hold. Then there exist two stationary points $\bar{u}_1 \neq \bar{u}_2$ with $J(\bar{u}_1) < J(\bar{u}_2)$. Now, (A) and (B) shown in the proof of Theorem 5.1 yield that there must exist an accumulation point \bar{u}_3 of $(u_{b(n)})_{n \in \mathbb{N}}$, i.e., a stationary point, with $J(\bar{u}_1) < J(\bar{u}_3) < J(\bar{u}_2)$. Iterating this procedure, we conclude that the set $\mathcal{N} \cap [J(\bar{u}_1), J(\bar{u}_2)]$ is dense in $[J(\bar{u}_1), J(\bar{u}_2)]$.

By continuity of $u \mapsto \mathcal{P}_{\mathcal{U}}(u - \tau \nabla J(u)) - u$ and compactness of \mathcal{U} , we see that

$$\mathcal{N} \cap [J(\bar{u}_1), J(\bar{u}_2)] = [J(\bar{u}_1), J(\bar{u}_2)],$$

contradicting our assumption that $\lambda(\mathcal{N}) = 0$.

For (R2), assume that $(u_{a(n)})_{n \in \mathbb{N}}$ has a non-stationary accumulation point \bar{u} . Since $\mathcal{S}(J)$ is compact, it holds

$$\text{dist}(\{\bar{u}\}, \mathcal{S}(J)) > 0.$$

Thus, by the same arguments as in Case 3.1, 3.2 and 3.3 within the proof of Theorem 5.1, we observe that such a point \bar{u} can not exist. \square

6 Backtracking

Choosing an appropriate step size in practice is usually very challenging. While a deterministic full gradient method is typically carried out with a line search scheme, it is unclear how such techniques can be included in standard stochastic gradient methods. However, since we have $\|\hat{G}_n - \nabla J(u_n)\| \rightarrow 0$ and $\|\hat{J}_n - J(u_n)\| \rightarrow 0$ in CSG, we can use these approximations to refine the step length by a backtracking line search method, similar to the deterministic case.

The stabilizing effect of these augmentations can be seen in Sect. 7.3, where we compare CSG with backtracking (Algorithm 3) to AdaGrad.

Definition 6.1 For simplicity, we define

$$s_n(t) := \mathcal{P}_{\mathcal{U}}(u_n - t\hat{G}_n).$$

Furthermore, given n gradient samples $\nabla_1 j(u_i, x_i)$ and n cost function samples $j(u_i, x_i)$, by calculating the weights $\alpha_i^{(n)}(u)$ w.r.t. a given point $u \in \mathcal{U}$, we define

$$\tilde{J}_n(u) = \sum_{i=1}^n \alpha_i^{(n)}(u) j(u_i, x_i) \quad \text{and} \quad \tilde{G}_n(u) = \sum_{i=1}^n \alpha_i^{(n)}(u) \nabla_1 j(u_i, x_i),$$

which are approximations to $J(u)$ and $\nabla J(u)$ respectively.

Based on the well known Armijo-Wolfe conditions from continuous optimization [50–52], we introduce the following step size conditions:

Definition 6.2 For $0 < c_1 < c_2 < 1$, we call $s_n(\tau_n)$ an Armijo step, if

$$\tilde{J}_n(s_n(\tau_n)) \leq \hat{J}_n - c_1 \hat{G}_n^\top (u_n - s_n(\tau_n)). \quad (\text{SW1})$$

Additionally, we Define the following Wolfe-type condition:

$$\tilde{G}_n(s_n(\tau_n))^\top (s_n(\tau_n) - u_n) \geq c_2 \hat{G}_n^\top (s_n(\tau_n) - u_n). \quad (\text{SW2})$$

We try to obtain a step size that satisfies (SW1) and (SW2) by a bisection approach, as formulated in Algorithm 2. Since we can not guarantee to find a suitable step size, we perform only a fixed number $T \in \mathbb{N}$ of backtracking steps. Notice that the curvature condition (SW2) only has an influence, if $u_n - \tau_n \hat{G}_n \in \mathcal{U}$ (see line 6 in Algorithm 2). This way, we gain the advantages of a Wolfe line search while inside \mathcal{U} , without ruling out stationary points at the boundary of \mathcal{U} .

For our convergence analysis, we assume that in each iteration of CSG with line search, Algorithm 2 is initiated with the same $\eta_n = \eta > 0$. From a practical point of view, we might also consider a diminishing sequence $(\eta_n)_{n \in \mathbb{N}}$ of backtracking initializations (see Sect. 7.3). The CSG method with backtracking line search (bCSG) is given in Algorithm 3. Since all of the terms $\tilde{J}_n(s_n(\tau_n))$, \hat{J}_n and \hat{G}_n appearing in

(SW1) contain some approximation error when compared to $J(s_n(\tau_n))$, $J(u_n)$ and $\nabla J(u_n)$ respectively, especially the first iterations of Algorithm 3 might profit from a slightly weaker formulation of (SW1). Therefore, in practice, we will replace (SW1) by the non-monotone version

$$\tilde{J}_n(s_n(\tau_n)) \leq \max_{k \in \{0, \dots, K\}} \hat{J}_{n-k} - c_1 \hat{G}_n^\top (u_n - s_n(\tau_n)), \quad (\text{SW1}^*)$$

for some $K \in \{0, \dots, n\}$.

Algorithm 2 Backtracking Refinement

Given $T \in \mathbb{N}$, $0 < c_1 < c_2 < 1$ appearing in (SW1*) and (SW2), $u_n \in \mathcal{U}$, and $\eta > 0$, set $t = 1$, $a = 0$, $b = \infty$, $\eta_A = \infty$.

while $t \leq T$ **do**

 Calculate step $s = \mathcal{P}_{\mathcal{U}}(u_n - \eta \hat{G}_n)$, weights $\alpha_k^{(n)}(s)$ and $\tilde{J}_n(s)$, $\tilde{G}_n(s)$

if (SW1*) is not satisfied **then**

$b = \eta$

else if $s = u_n - \eta \hat{G}_n$ **and** (SW2) is not satisfied **then**

$a = \eta$

$\eta_A = \eta$

else

break

end if

if $b < \infty$ **then**

$\eta = \frac{a+b}{2}$

else

$\eta = 2a$

end if

$t \leftarrow t + 1$

end while

if $t = T + 1$ **and** $\eta_A < \infty$ **then**

$\tau_n = \eta_A$

else

$\tau_n = \eta$

end if

6.1 Convergence results

For CSG with backtracking line search, we obtain the same convergence results as for constant step sizes:

Theorem 6.1 (Convergence for Backtracking Line Search) *Let $\mathcal{S}(J)$ be the set of stationary points of J on U as defined in Definition 4.1. Assume that*

$$\mathcal{N} := \{J(u) : u \in \mathcal{S}(J)\} \subseteq \mathbb{R}$$

is of Lebesgue-measure zero and T in Algorithm 2 is chosen large enough, such that $2^{-T} \eta < \frac{2}{L}$. Then, with probability 1, every accumulation point of the sequence

Algorithm 3 Backtracking CSG (bCSG)

```

1: Given  $u_0 \in \mathcal{U}$ , and a positive sequence  $(\eta_n)_{n \in \mathbb{N}}$ ,
2: while Termination condition not met do
3:   Sample objective function:
4:    $j_n = j(u_n, x_n)$ 
5:   Sample gradient:
6:    $g_n = \nabla_1 j(u_n, x_n)$ 
7:   Calculate weights  $\alpha_k$ 
8:   Calculate search direction:
9:    $\hat{G}_n = \sum_{k=1}^n \alpha_k g_k$ 
10:  Compute objective function value approximation:
11:   $\hat{J}_n = \sum_{k=1}^n \alpha_k j_k$ 
12:  Calculate step size  $\tau_n$  by Algorithm 2 with start at  $\eta_n$ 
13:  Gradient step:
14:   $u_{n+1} = \mathcal{P}_{\mathcal{U}}(u_n - \tau_n \hat{G}_n)$ 
15:  Update index:
16:   $n \leftarrow n + 1$ 
17: end while

```

$(u_n)_{n \in \mathbb{N}}$ generated by Algorithm 3 is stationary and we have convergence in function values.

If J satisfies the stronger assumption of having only finitely many stationary points, $(u_n)_{n \in \mathbb{N}}$ converges to a stationary point of J .

Proof Using the same notation as in the proof of Theorem 5.1, we again assume the fixed sample sequence $(x_n)_{n \in \mathbb{N}}$ and all of its subsequences constructed in the following to be dense in $\text{supp}(\mu)$, which holds with probability 1, see Remark 2.1. Notice first, that there are only two possible outcomes of Algorithm 2: Either τ_n satisfies (SW1), or $\tau_n = 2^{-T}\eta < \frac{2}{L}$. Furthermore, as we have seen in the proof of Lemma 4.3, for all $\varepsilon > 0$ almost all u_n lie in ε -Balls around the accumulation points of $(u_n)_{n \in \mathbb{N}}$, since $(u_n)_{n \in \mathbb{N}}$ is bounded. Therefore, $\tilde{J}_n(u_{n+1}) - J(u_{n+1}) \rightarrow 0$ and $\tilde{G}_n(u_{n+1}) - \nabla J(u_{n+1}) \rightarrow 0$ (compare Lemma 4.6). Since we already know that the steps with constant step size $\tau_n = 2^{-T}\eta < \frac{2}{L}$ can be split in descent steps and steps which satisfy $\|u_{n+1} - u_n\| \rightarrow 0$, we now take a closer look at the Armijo-steps, i.e., steps with $\tau_n \neq 2^{-T}\eta$.

If $\tau_n \neq 2^{-T}\eta$, by (SW1) and Lemma 4.2, it holds

$$\begin{aligned}
 J(u_{n+1}) - J(u_n) &\leq -c_1 \frac{\|u_{n+1} - u_n\|_{\mathcal{U}}^2}{\tau_n^2} + \left| J(u_n) - \hat{J}_n \right| + \left| J(u_{n+1}) - \tilde{J}_n(u_{n+1}) \right| \\
 &\leq -c_1 \frac{\|u_{n+1} - u_n\|_{\mathcal{U}}^2}{\tau_{\max}^2} + \left| J(u_n) - \hat{J}_n \right| + \left| J(u_{n+1}) - \tilde{J}_n(u_{n+1}) \right|.
 \end{aligned}$$

Therefore, we either have

$$\left| J(u_n) - \hat{J}_n \right| + \left| J(u_{n+1}) - \tilde{J}_n(u_{n+1}) \right| \leq c_1 \frac{\|u_{n+1} - u_n\|_{\mathcal{U}}^2}{\tau_{\max}^2},$$

in which case it holds $J(u_{n+1}) \leq J(u_n)$, or

$$\left| J(u_n) - \hat{J}_n \right| + \left| J(u_{n+1}) - \tilde{J}_n(u_{n+1}) \right| > c_1 \frac{\|u_{n+1} - u_n\|_{\mathcal{U}}^2}{\tau_{\max}^2},$$

in which case $\tilde{J}_n(u_{n+1}) - J(u_{n+1}) \rightarrow 0$ and $\hat{J}_n - J(u_n) \rightarrow 0$ yield $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$.

Thus, regardless of whether or not $\tau_n = 2^{-T} \eta$, we can split $(u_n)_{n \in \mathbb{N}}$ in a subsequence of descent steps and a subsequence of steps with $\|u_{n+1} - u_n\|_{\mathcal{U}} \rightarrow 0$. The rest of the proof is now identical to the proof of Theorem 5.1 and Theorem 5.2. \square

6.2 Estimations for the lipschitz constant of ∇J

We have already seen, that the Lipschitz constant L of ∇J is closely connected with efficient bounds on the step sizes. However, in general, we can not expect to have any knowledge of L a priori. Thus, we are interested in an approximation of L , that can be calculated during the optimization process.

Investigating the proof of Lemma 4.1 in [44]

$$\begin{aligned} J(u_1) &= J(u_2) + \int_0^1 \langle \nabla J(u_2 + t(u_1 - u_2)), u_1 - u_2 \rangle dt \\ &= J(u_2) + \langle \nabla J(u_2), u_1 - u_2 \rangle \\ &\quad + \int_0^1 \langle \nabla J(u_2 + t(u_1 - u_2)) - \nabla J(u_2), u_1 - u_2 \rangle dt \\ &\leq J(u_2) + \langle \nabla J(u_2), u_1 - u_2 \rangle \\ &\quad + \int_0^1 \|\nabla J(u_2 + t(u_1 - u_2)) - \nabla J(u_2)\| \cdot \|u_1 - u_2\|_{\mathcal{U}} dt \\ &\leq J(u_2) + \langle \nabla J(u_2), u_1 - u_2 \rangle + \int_0^1 L t \|u_1 - u_2\|_{\mathcal{U}}^2 dt \\ &= J(u_2) + \langle \nabla J(u_2), u_1 - u_2 \rangle + \frac{L}{2} \|u_1 - u_2\|_{\mathcal{U}}^2, \end{aligned}$$

we observe that we do not need the true Lipschitz constant L of ∇J for the second inequality. Instead, it is sufficient to choose any constant $C = C(u_1, u_2)$ that satisfies

$$\|\nabla J(u_2 + t(u_1 - u_2)) - \nabla J(u_2)\| \leq C \|u_1 - u_2\|_{\mathcal{U}} \quad \text{for all } t \in [0, 1].$$

To motivate our approach, assume that J is twice continuously differentiable. In this case, a possible approximation to the constant C_n in iteration n is $\|\nabla^2 J(u_n)\|$. Therefore, utilizing the previous gradient approximations, we obtain

$$C_n \approx \|\nabla^2 J(u_n)\| \approx \frac{\|\nabla J(u_n) - \nabla J(u_{n-1})\|}{\|u_n - u_{n-1}\|_{\mathcal{U}}} \approx \frac{\|\hat{G}_n - \hat{G}_{n-1}\|}{\|u_n - u_{n-1}\|_{\mathcal{U}}}.$$

Then, C_n^{-1} yields a good initial step size for our backtracking line search. To circumvent high oscillation of C_n , which may arise from the approximation errors of the terms involved, we project C_n onto the interval $[C_{\min}, C_{\max}] \subseteq \mathbb{R}$, where $0 < C_{\min} < C_{\max} < \frac{2^{T+1}}{L}$, i.e.,

$$C_n = \min \left\{ C_{\max}, \max \left\{ C_{\min}, \frac{\|\hat{G}_n - \hat{G}_{n-1}\|}{\|u_n - u_{n-1}\|_{\mathcal{U}}} \right\} \right\}. \quad (19)$$

If possible, C_{\min} and C_{\max} should be chosen according to information concerning L . However, tight bounds on these quantities are not needed, as long as T is chosen large enough. The resulting SCIBL-CSG (SCaling Independent Backtracking Line search) method is presented in Algorithm 4. Notice that SCIBL-CSG does not require any a priori choice of step sizes and yields the same convergence results as bCSG.

Algorithm 4 SCIBL-CSG

Given $u_0 \in \mathcal{U}$,
while Termination condition not met **do**
 Sample objective function:
 $j_n = j(u_n, x_n)$
 Sample gradient:
 $g_n = \nabla_1 j(u_n, x_n)$
 Calculate weights α_k
 Calculate search direction:
 $\hat{G}_n = \sum_{k=1}^n \alpha_k g_k$
 Compute objective function value approximation:
 $\hat{j}_n = \sum_{k=1}^n \alpha_k j_k$
 Calculate C_n by (19).
 Calculate step size τ_n by Algorithm 2 with start at $\frac{1}{C_n}$
 Gradient step:
 $u_{n+1} = \mathcal{P}_{\mathcal{U}}(u_n - \tau_n \hat{G}_n)$
 Update index:
 $n \leftarrow n + 1$
end while

7 Examples to illustrate key aspects of CSG performance

So far, all results presented indicate promising features of CSG in theory, but did not provide any insight on the actual practical performance. While a detailed numerical analysis of CSG can be found in [27], we want to investigate isolated key features with the help of some academic examples.

To be precise, we start by comparing the performances of CSG and standard SG for constant step sizes (Sect. 7.1). Afterwards, the generalized setting, mentioned in Remark 4.1, is explored for a composite objective function (Sect. 7.2). Note that we consider only the case of two nested expected values, because for more complex

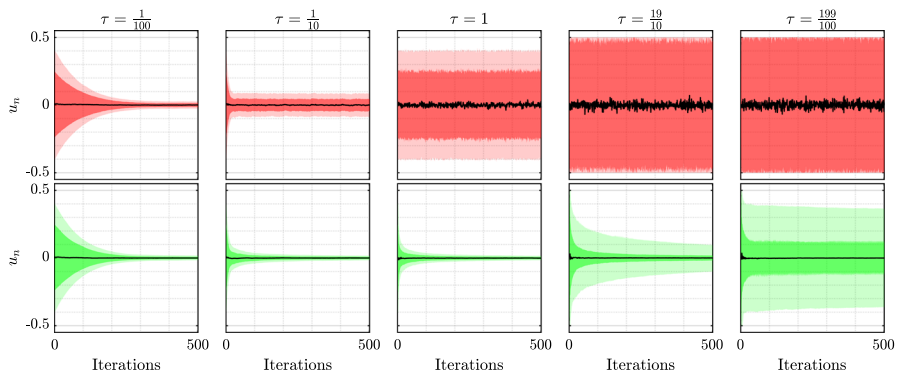


Fig. 10 Comparison of the iterates produced by 500 steps of SG (red/first row) and CSG (green/second row) with 2000 random starting points $u_0 \in \mathcal{U}$. Both methods have been tested for the five different constant step sizes $\tau \in \{0.01, 0.1, 1, 1.9, 1.99\}$ (first to fifth column). The shaded areas indicate the quantiles $P_{0.1,0.9}$ (light) and $P_{0.25,0.75}$ (dark), while the solid line represents the median of the 2000 runs (Color figure online)

settings, there is a lack of efficient stochastic optimization techniques in the literature. Lastly, the line search techniques presented in Sect. 6 are tested and compared to the adaptive step sizes chosen in AdaGrad (Sect. 7.3).

7.1 Academic example for constant step size

Define $\mathcal{U} = [-\frac{1}{2}, \frac{1}{2}]$, $\mathcal{X} = [-\frac{1}{2}, \frac{1}{2}]$ and consider the problem

$$\min_{u \in \mathcal{U}} \frac{1}{2} \int_{\mathcal{X}} (u - x)^2 dx. \quad (20)$$

It is easy to see that (20) has a unique solution $u^* = 0$. Furthermore, the objective function is L -smooth (with Lipschitz constant 1) and even strictly convex. Thus, by Theorem 5.1, the CSG method with a constant positive step size $\tau < 2$ produces a sequence $(u_n)_{n \in \mathbb{N}}$ that satisfies $u_n \rightarrow 0$.

However, even in this highly regular setting, the commonly used basic SG method does not guarantee convergence of the iterates for a constant step size.

To demonstrate this behavior of both CSG and SG, we draw 2000 random starting points $u_0 \in \mathcal{U}$ and compare the iterates produced by CSG and SG with five different constant step sizes ($\tau \in \{0.01, 0.1, 1, 1.9, 1.99\}$). The CSG integration weights were calculated using the empirical method, i.e., the computationally cheapest choice. The results are shown in Fig. 10.

As expected, the iterates produced by the SG method do not converge to the optimal solution, but instead remain in a neighborhood of u^* . The radius of said neighborhood depends on the choice of τ and decreases for smaller τ , see [53, Theorem 4.6].

7.2 Example for a composite objective function

To study the performance of CSG in the generalized setting, we consider an optimization problem in which the objective function is not of the type (1), but instead falls in the broader class of possible settings mentioned in Remark 4.1. Thus, we introduce the sets $\mathcal{U} = [0, 10]$, $\mathcal{X} = [-1, 1]$ and $\mathcal{Y} = (-3, 3)$ and define the optimization problem

$$\min_{u \in \mathcal{U}} \frac{1}{20} \int_{\mathcal{Y}} \left(2y + 5 \int_{\mathcal{X}} \cos\left(\frac{u-x}{\pi}\right) dx \right)^2 dy. \quad (21)$$

The optimal solution $u^* = \frac{\pi^2}{2}$ to (21) can be found analytically. The nonlinear fashion in which the inner integral over \mathcal{X} enters the objective function prohibits us from using SG-type methods to solve (21). There is, however, the possibility to use the stochastic compositional gradient descent method (SCGD), which was proposed in [26] and is specifically designed for optimization problems of the form (21). Each SCGD iteration consists of two main steps: The inner integral is approximated by samples using iterative updating with a slow-diminishing step size. This approximation is then used to carry out a stochastic gradient descent step with a fast-diminishing step size.

For numerical comparison, we choose 1000 random starting points in $[\frac{11}{2}, \frac{19}{2}]$, i.e., the right half of \mathcal{U} . In our tests, the accelerated SCGD method (see [26]) performed better than basic SCGD, mainly since the objective function of (21) is strongly convex in a neighborhood of u^* . Thus, we compare the results obtained by CSG to the aSCGD algorithm, for which we chose the optimal step sizes according to [26, Theorem 7]. For CSG, we chose a constant step size $\tau = \frac{1}{30}$, which represents a rough approximation to the inverse of the Lipschitz constant L . The results are given in Fig. 11.

Furthermore, we are interested in the number of steps each method has to perform such that the distance to u^* lies (and stays) within a given tolerance. Thus, we also analyzed the number of steps after which the different methods obtain a result within a tolerance of 10^{-1} in at least 90% of all runs. The results are shown in Fig. 12.

7.3 Step size stability for bCSG

To analyze the proclaimed stability of bCSG with respect to the initially guessed step size η_n , we set $\mathcal{U} = [-10, 10]^5$, $\mathcal{X} = [-1, 1]^5$ and consider the Problem

$$\min_{u \in \mathcal{U}} J(u), \quad (22)$$

where

$$J(u) = - \int_{\mathcal{X}} \frac{20}{1 + \|u - x\|^2} dx.$$

Problem (22) has the unique solution $u^* = 0 \in \mathcal{U}$, which can be found analytically.

Fig. 11 Absolute error $\|u_n - u^*\|$ during the optimization process. From top to bottom: aSCGD (red), CSG with empirical weights (cyan), CSG with inexact hybrid weights with $f(n) = \lfloor n^{1.5} \rfloor$ (green) and CSG with exact hybrid weights (blue). The shaded areas indicate the quantiles $P_{0.1,0.9}$ (light) and $P_{0.25,0.75}$ (dark) (Color figure online)

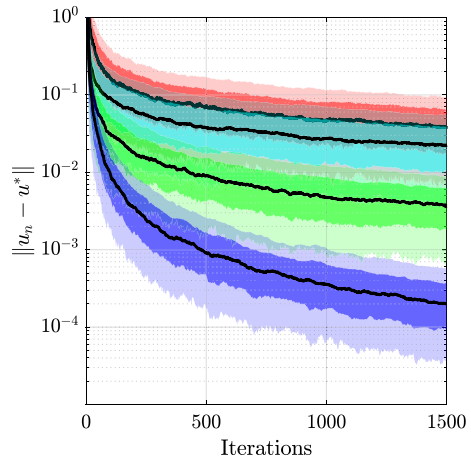
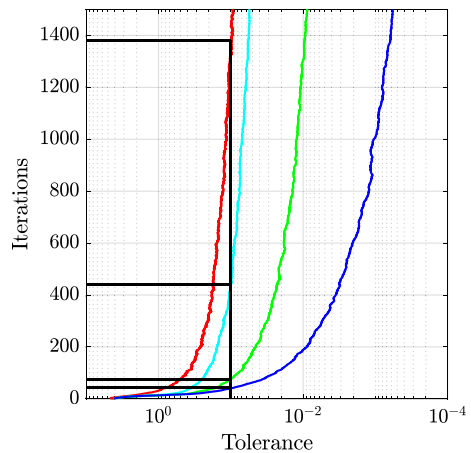


Fig. 12 Minimum number of steps needed for the different algorithms such that at least 90% of the runs achieve an absolute error smaller than the given tolerance. the colors are chosen in the same fashion as in Fig. 11. The exact numbers for a tolerance of 10^{-1} are 42 (exact hybrid), 76 (inexact hybrid), 440 (empirical) and 1382 (aSCGD)



As a comparison to our method, we choose the AdaGrad [25] algorithm, as it is widely used for problems of type (1). Both AdaGrad and bCSG start each iteration with a prescribed step size $\eta_n > 0$, based on which the calculation of the true step size τ_n is performed (see Algorithm 2). We want to test the stability of both methods with respect to the initially chosen step length. For this purpose, we set $\eta_n = \frac{\tau_0}{n^d}$, where $\tau_0 > 0$, n is the iteration count and $d \in [0, 1]$ is fixed.

For each combination of τ_0 and d , we choose 1200 random starting points in \mathcal{U} and perform 500 optimization steps with both AdaGrad and backtracking CSG. Again, the integration weight calculation in bCSG was carried out using the empirical method, leading to a faster weight calculation while decreasing the overall progress per iteration performance. The median of the absolute error $\|u_{500} - u^*\|$ after the optimization, depending on d and τ_0 , is presented in Fig. 13.

While there are a few instances where AdaGrad yields a better result than backtracking CSG, we observe that the performance of AdaGrad changes rapidly, especially with

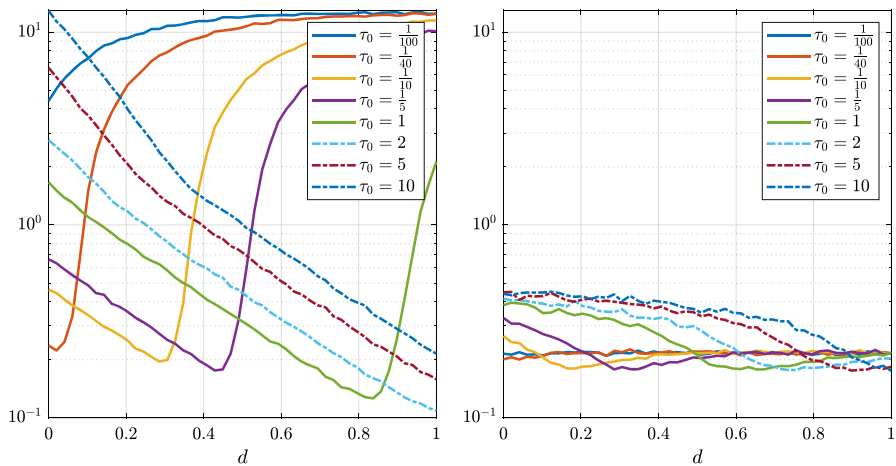


Fig. 13 Median final error $\|u_{500} - u^*\|$ after 500 iterations of AdaGrad (left) and bCSG (right). Depending on the constants $\tau_0 > 0$ and $d \in [0, 1]$, the initially chosen step size in each iteration is given by $\eta_n = \tau_0 n^{-d}$

respect to the parameter d . The backtracking CSG method on the other hand performs superior in most cases and is much less dependent on the choice of parameters.

8 Conclusion and outlook

In this contribution, we provided a detailed convergence analysis of the CSG method. The calculation of the integration weights was enhanced by several new approaches, which have been discussed and generalized for the possible implementation of mini-batches.

We provided a convergence proof for the CSG method when carried out with a small enough constant step size. Additionally, it was shown that CSG can be augmented by an Armijo-type backtracking line search, based on the gradient and objective function approximations generated by CSG in the course of the iterations. The resulting bCSG scheme was proven to converge under mild assumptions and was shown to yield stable results for a large spectrum of hyperparameters. Lastly, we combined a heuristic approach for approximating the Lipschitz constant of the gradient with bCSG to obtain a method that requires no a priori step size rule and almost no information about the optimization problem.

For all CSG variants, the stated convergence results are similar to convergence results for full gradient schemes, i.e., every accumulation point of the sequence of iterates is stationary and we have convergence in objective function values. Furthermore, as is the case for full gradient methods, if the optimization problem has only finitely many stationary points, the presented CSG variants produce a sequence which is guaranteed to converge to one of these stationary points.

However, none of the presented convergence results for CSG give any indication of the underlying rate of convergence. Furthermore, while the performance of all

proposed CSG variants was tested on academic examples, it is important to analyze how they compare to algorithms from literature and commercial solvers, when used in real world applications.

Detailed numerical results concerning both of these aspects can be found in [27].

Acknowledgements The research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)-Project-ID 416229255-CRC 1411).

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability In this contribution, only simple academic examples are used to visualize the theoretical results. These can be reproduced based on the given algorithms. Nevertheless, the simulation datasets generated during the current study are available from the corresponding author on request.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Pflug, L., Bernhardt, N., Grieshammer, M., Stingl, M.: CSG: a new stochastic gradient method for the efficient solution of structural optimization problems with infinitely many states. *Struct. Multidiscip. Optim.* **61**(6), 2595–2611 (2020). <https://doi.org/10.1007/s00158-020-02571-x>
2. Kim, C., Lee, J., Yoo, J.: Machine learning-combined topology optimization for functionary graded composite structure design. *Comput. Methods Appl. Mech. Eng.* **387**, 114158–32 (2021). <https://doi.org/10.1016/j.cma.2021.114158>
3. Evstatiev, E.G., Finn, J.M., Shadwick, B.A., Hengartner, N.: Noise and error analysis and optimization in particle-based kinetic plasma simulations. *J. Comput. Phys.* **440**, 110394–28 (2021). <https://doi.org/10.1016/j.jcp.2021.110394>
4. Wadbro, E., Berggren, M.: Topology optimization of an acoustic horn. *Comput. Methods Appl. Mech. Eng.* **196**(1–3), 420–436 (2006). <https://doi.org/10.1016/j.cma.2006.05.005>
5. Hassan, E., Wadbro, E., Berggren, M.: Topology optimization of metallic antennas. *IEEE Trans. Antennas Propag.* **62**(5), 2488–2500 (2014). <https://doi.org/10.1109/TAP.2014.2309112>
6. Semmler, J., Pflug, L., Stingl, M., Leugering, G.: Shape optimization in electromagnetic applications. In: *New Trends in Shape Optimization. Internat. Ser. Numer. Math.*, vol. 166, pp. 251–269. Birkhäuser/Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17563-8_11
7. Singh, S., Pflug, L., Mergheim, J., Stingl, M.: Robust design optimization for enhancing delamination resistance of composites. *Internat. J. Numer. Methods Eng.* **124**(6), 1381–1404 (2023). <https://doi.org/10.1002/nme.7168>
8. Martin, M., Nobile, F.: Pde-constrained optimal control problems with uncertain parameters using saga. *SIAM/ASA J. Uncertain. Quanti.* **9**(3), 979–1012 (2021). <https://doi.org/10.1137/18M1224076>
9. Borzi, A., von Winckel, G.: Multigrid methods and sparse-grid collocation techniques for parabolic optimal control problems with random coefficients. *SIAM J. Sci. Comput.* **31**(3), 2172–2192 (2009). <https://doi.org/10.1137/070711311>

10. Babuška, I., Nobile, F., Tempone, R.: A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Rev.* **52**(2), 317–355 (2010). <https://doi.org/10.1137/100786356>
11. Babuska, I., Tempone, R., Zouraris, G.E.: Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM J. Numer. Anal.* **42**(2), 800–825 (2004). <https://doi.org/10.1137/S0036142902418680>
12. Geiersbach, C., Pflug, G.C.: Projected stochastic gradients for convex constrained problems in Hilbert spaces. *SIAM J. Optim.* **29**(3), 2079–2099 (2019). <https://doi.org/10.1137/18M1200208>
13. Geiersbach, C., Wollner, W.: A stochastic gradient method with mesh refinement for pde-constrained optimization under uncertainty. *SIAM J. Sci. Comput.* **42**(5), 2750–2772 (2020). <https://doi.org/10.1137/19M1263297>
14. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer Series in Operations Research, p. 636. Springer, New York (1999). <https://doi.org/10.1007/b98874>
15. Pflug, G.C., Pichler, A.: Multistage Stochastic Optimization. Springer Series in Operations Research and Financial Engineering, p. 301. Springer, Cham (2014). <https://doi.org/10.1007/978-3-319-08843-3>
16. Robbins, H., Monro, S.: A stochastic approximation method. *Ann. Math. Stat.* **22**, 400–407 (1951). <https://doi.org/10.1214/aoms/1177729586>
17. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. *Math. Program.* **162**(1–2), 83–112 (2017). <https://doi.org/10.1007/s10107-016-1030-6>
18. Curtis, F.E., O'Neill, M.J., Robinson, D.P.: Worst-case complexity of an SQP method for nonlinear equality constrained stochastic optimization. arXiv preprint [arXiv:2112.14799](https://arxiv.org/abs/2112.14799) (2021). <https://doi.org/10.48550/arXiv.2112.14799>
19. Berahas, A.S., Curtis, F.E., Robinson, D., Zhou, B.: Sequential quadratic optimization for nonlinear equality constrained stochastic optimization. *SIAM J. Optim.* **31**(2), 1352–1379 (2021). <https://doi.org/10.1137/20M1354556>
20. Bordes, A., Bottou, L., Gallinari, P.: SGD-QN: careful quasi-Newton stochastic gradient descent. *J. Mach. Learn. Res.* **10**, 1737–1754 (2009)
21. Pilanci, M., Wainwright, M.J.: Newton sketch: a near linear-time optimization algorithm with linear-quadratic convergence. *SIAM J. Optim.* **27**(1), 205–245 (2017). <https://doi.org/10.1137/15M1021106>
22. Byrd, R.H., Hansen, S.L., Nocedal, J., Singer, Y.: A stochastic quasi-Newton method for large-scale optimization. *SIAM J. Optim.* **26**(2), 1008–1031 (2016). <https://doi.org/10.1137/140954362>
23. Moritz, P., Nishihara, R., Jordan, M.: A linearly-convergent stochastic l-bfgs algorithm. In: Artificial Intelligence and Statistics, pp. 249–258 (2016). PMLR
24. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014). <https://doi.org/10.48550/arXiv.1412.6980>
25. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.* **12**, 2121–2159 (2011)
26. Wang, M., Fang, E.X., Liu, H.: Stochastic compositional gradient descent: algorithms for minimizing compositions of expected-value functions. *Math. Program.* **161**(1–2), 419–449 (2017). <https://doi.org/10.1007/s10107-016-1017-3>
27. Grieshammer, P., Pflug, L., Stingl, M., Uihlein, A.: The continuous stochastic gradient method: part II—application and numerics. *Comput. Optim. Appl.* (2023). <https://doi.org/10.1007/s10589-023-00540-w>
28. Zhao, Y., Xie, Z., Gu, H., Zhu, C., Gu, Z.: Bio-inspired variable structural color materials. *Chem. Soc. Rev.* **41**, 3297–3317 (2012). <https://doi.org/10.1039/C2CS15267C>
29. Wang, J., Sultan, U., Goerlitzer, E.S.A., Mbah, C.F., Engel, M.S., Vogel, N.: Structural color of colloidal clusters as a tool to investigate structure and dynamics. In: Advanced Functional Materials, vol. 30 (2019)
30. England, G.T., Russell, C., Shirman, E., Kay, T., Vogel, N., Aizenberg, J.: The optical Janus effect: asymmetric structural color reflection materials. *Adv. Mater.* (2017). <https://doi.org/10.1002/adma.201606876>
31. Xiao, M., Hu, Z., Wang, Z., Li, Y., Tormo, A.D., Thomas, N.L., Wang, B., Gianneschi, N.C., Shawkey, M.D., Dhinojwala, A.: Bioinspired bright noniridescent photonic melanin supraballs. *Sci. Adv.* **3**(9), 1701151 (2017). <https://doi.org/10.1126/sciadv.1701151>
32. Goerlitzer, E.S.A., Klupp Taylor, R.N., Vogel, N.: Bioinspired photonic pigments from colloidal self-assembly. *Adv. Mater.* **30**(28), 1706654 (2018). <https://doi.org/10.1002/adma.201706654>

33. Uihlein, A., Pflug, L., Stingl, M.: Optimizing color of particulate products. *PAMM* **22**(1), 202200047 (2023). <https://doi.org/10.1002/pamm.202200047>
34. Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *J. Basic Eng.* **86**(1), 97–106 (1964). <https://doi.org/10.1115/1.3653121>
35. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. vol. 13, pp. 455–492 (1998). <https://doi.org/10.1023/A:1008306431147>. Workshop on Global Optimization (Trier, 1997)
36. Audet, C., Dennis, J.: Analysis of generalized pattern searches. *SIAM J. Optim.* (2000). <https://doi.org/10.1137/S1052623400378742>
37. Klenke, A.: Probability Theory. Universitext, p. 616. Springer, London (2008). <https://doi.org/10.1007/978-1-84800-048-3>. A comprehensive course, Translated from the 2006 German original
38. Burrough, P., McDonnell, R., Lloyd, C.: 8.11 nearest neighbours: Thiessen (dirichlet/voronoi) polygons. *Principles of Geographical Information Systems* (2015)
39. Dudley, R.M.: Central limit theorems for empirical measures. *Ann. Probab.* (no. 6) 899–929 (1978)
40. Varadarajan, V.S.: On the convergence of sample probability distributions. *Sankhyā* **19**, 23–26 (1958)
41. Folland, G.B.: A Guide to Advanced Real Analysis. The Dolciani Mathematical Expositions, vol. 37, p. 107. Mathematical Association of America, Washington DC (2009). MAA Guides, 2
42. Goldstein, A.A.: Convex programming in Hilbert space. *Bull. Am. Math. Soc.* **70**, 709–710 (1964). <https://doi.org/10.1090/S0002-9904-1964-11178-2>
43. Levitin, E.S., Polyak, B.T.: Constrained minimization methods. *USSR Comput. Math. Math. Phys.* **6**(5), 1–50 (1966). [https://doi.org/10.1016/0041-5553\(66\)90114-5](https://doi.org/10.1016/0041-5553(66)90114-5)
44. Beck, A.: First-order Methods in Optimization. MOS-SIAM Series on Optimization, vol. 25, p. 475. Society for Industrial and Applied Mathematics (SIAM); Mathematical Optimization Society, Philadelphia (2017). <https://doi.org/10.1137/1.9781611974997.ch1>
45. Gibbs, A.L., Su, F.E.: On choosing and bounding probability metrics. *Int. Stat. Rev.* **70**(3), 419–435 (2002)
46. Sard, A.: The measure of the critical values of differentiable maps. *Bull. Am. Math. Soc.* **48**, 883–890 (1942). <https://doi.org/10.1090/S0002-9904-1942-07811-6>
47. Guillemin, V., Pollack, A.: Differential Topology, p. 222. Prentice-Hall Inc, Englewood Cliffs (1974)
48. Whitney, H.: A function not constant on a connected set of critical points. *Duke Math. J.* **1**(4), 514–517 (1935). <https://doi.org/10.1215/S0012-7094-35-00138-7>
49. Kaufman, R.: A singular map of a cube onto a square. *J. Differ. Geom.* **14**(4), 593–594 (1979)
50. Armijo, L.: Minimization of functions having Lipschitz continuous first partial derivatives. *Pac. J. Math.* **16**, 1–3 (1966)
51. Wolfe, P.: Convergence conditions for ascent methods. *SIAM Rev.* **11**, 226–235 (1969). <https://doi.org/10.1137/1011036>
52. Wolfe, P.: Convergence conditions for ascent methods. II. Some corrections. *SIAM Rev.* **13**, 185–188 (1971). <https://doi.org/10.1137/1013035>
53. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. *SIAM Rev.* **60**(2), 223–311 (2018). <https://doi.org/10.1137/16M1080173>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.