



## Simulation model calibration with dynamic stratification and adaptive sampling

Pranav Jain, Sara Shashaani & Eunshin Byon

To cite this article: Pranav Jain, Sara Shashaani & Eunshin Byon (01 Nov 2024): Simulation model calibration with dynamic stratification and adaptive sampling, Journal of Simulation, DOI: [10.1080/17477778.2024.2420807](https://doi.org/10.1080/17477778.2024.2420807)

To link to this article: <https://doi.org/10.1080/17477778.2024.2420807>



[View supplementary material](#)



Published online: 01 Nov 2024.



[Submit your article to this journal](#)



Article views: 3



[View related articles](#)




[View Crossmark data](#)

RESEARCH ARTICLE



# Simulation model calibration with dynamic stratification and adaptive sampling

Pranav Jain<sup>a</sup>, Sara Shashaani <sup>a</sup> and Eunshin Byon<sup>b</sup>

<sup>a</sup>Edward P. Fitts Department of Industrial and System Engineering, North Carolina State University, Raleigh, NC, USA; <sup>b</sup>Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, USA

## ABSTRACT

Calibrating simulation models that take large quantities of multi-dimensional data as input is a hard simulation optimization problem. Existing adaptive sampling strategies offer a methodological solution. However, they may not sufficiently reduce the computational cost for estimation and solution algorithm's progress within a limited budget due to extreme noise levels and heteroscedasticity of system responses. We propose integrating stratification with adaptive sampling for the purpose of efficiency in optimization. Stratification can exploit local dependence in the simulation inputs and outputs. Yet, the state-of-the-art does not provide a full capability to adaptively stratify the data as different solution alternatives are evaluated. We devise two procedures for data-driven calibration problems that involve a large dataset with multiple covariates to calibrate models within a fixed overall simulation budget. The first approach dynamically stratifies the input data using binary trees, while the second approach uses closed-form solutions based on linearity assumptions between the objective function and concomitant variables. We find that dynamical adjustment of stratification structure accelerates optimization and reduces run-to-run variability in generated solutions. Our case study for calibrating a wind power simulation model, widely used in the wind industry, using the proposed stratified adaptive sampling, shows better-calibrated parameters under a limited budget.

## ARTICLE HISTORY

Received 10 January 2024  
Accepted 18 October 2024

## KEYWORDS

Data-driven calibration;  
heteroscedasticity;  
post-stratification;  
trust-region optimization

## 1. Introduction

Many simulation models, particularly those used to emulate real engineering systems, have physically unobservable parameters. This can be due to having a simulation model that has simplified the real system's dynamics (similar to low-fidelity models), or due to the environmental characteristics in which the real system operates (for example, the geographical and weather-related effects for a particular location and time), or due to the need for removing the initial transient state of the simulation to prepare the simulated data for analysis in time-dependent simulation outputs (for example, the warm-up period for a steady-state analysis). Particularly, in the first two examples, differences between simulated and real data are addressed during the important practice of calibration (Sargent, 2010; Schruben, 1980).

Calibration of simulation experiments with real-world observations is generally done through meta-modeling approaches, typically with Bayesian models (Kennedy & O'Hagan, 2001; Pousi et al., 2013). However, these methods do not scale well with the size of the data or the number of the calibration parameters (Jeong & Byon, 2024; Jeong et al., 2023). In the simulation literature, the so-called *direct model*

*calibration* is one that formulates the problem as a simulation optimization where the empirical loss is minimized by *searching for the optimal* calibration parameters over their feasible space (Tolk et al., 2017) [Chapter 3]. *Global search methods such as simulated annealing (SA) or random search (RS), as well as meta-heuristics such as the genetic algorithm or particle swarm optimization* are popular for such problems (Guzmán-Cruz et al., 2009; Juan Felipe Parra & Arango-Aramburo, 2018; S. Liu et al., 2007; Tahmasebi et al., 2012) with the downside of lacking computational efficiency and convergence guarantees. Efficient stochastic optimization methods have proven effective for *simulation model calibration* using stochastic gradient (B. Liu et al., 2022) or line search methods (J. Yuan et al., 2012).

### 1.1. Calibration as a simulation optimization

In traditional calibration of stochastic simulations, the discrepancy between the simulated and observed output values is computed while the inputs are simulated following an input probability model – a common practice in calibrating epidemiological models (Cheng et al., 2023). But in many simulation

experiments, the output data as well as real (not simulated) input data is used. We term these calibration problems *data-driven calibration* given that the input itself is directly queried from a real dataset of *both inputs and outputs of the real-world system*.

For example, in wind power generation models, a simulation model depends on unobservable parameters, such as the *wake* parameter that describes the effect of wind decay in downstream turbines. The wake parameter  $\theta$  can depend on a wind farm’s location and other local characteristics. Suppose that  $((x_i, y_i) : i = 1, 2, \dots, n)$  is a collection of observed wind characteristic vectors  $x_i \in \mathbb{R}^l$  (inputs), and observed generated wind power  $y_i \in \mathbb{R}^1$  (outputs) from a particular wind farm over time, where  $p$  is the number of turbines in the wind farm. The simulation model generates, as output, the predicted power – a vector-valued function  $h(\theta, x_i)$  in  $\mathbb{R}^1$ . The calibration problem involves finding the wake parameter that best matches simulated and real outputs, i.e.,

$$\min_{\theta \in \mathbb{R}} \sum_{i=1}^n \|h(\theta, x_i) - y_i\|_2^2. \quad (1)$$

*The problem above is also known as the empirical risk minimization (ERM). The simulation model, if accurately tuned, can be used to make decisions about the real system. However, if  $n$  is large, then enumerating all  $n$  data points to evaluate the performance under each  $\theta$  will be very inefficient because each  $h(\theta, x_i)$  evaluation requires running the simulation. In this case, one might resort to sampling, which renders ERM a stochastic problem. Using randomly selected small-scale data can alleviate the computational burden, so long as we can accurately tune the simulation model without utilizing all the observed data points. But when data is noisy, small samples result in inaccurate inference about the calibration parameter. In many applications such as those involving reliability (e.g., in wind power generation), using suboptimal calibration parameters to make decisions about the real system can cause high-risk consequences.*

## 1.2. Contributions

Viewing calibration as a simulation-optimization process, where the expected discrepancy between the model and real data is minimized, raises the question: can one reduce the algorithm complexity, that is, the overall simulation model runs to find a robust calibration parameter? Often, under a pre-specified computational budget, answering this question concerns allocating effort for (i) exploration, (ii) exploitation, and (iii) estimation in each iteration of the optimization algorithm (Andradóttir & Prudius, 2009; Gao et al., 2015). Better exploration of the calibration parameter space and finding near-optimal solutions

requires reducing the per-iteration budget, that is, the cost of estimation, as much as possible. We will formalize this discussion with a survey on adaptive sampling stochastic optimization methods used towards this goal in Section 2. But what is unique about a calibration problem to obtain cheaper estimates throughout the optimization process?

Ample input and output data in calibration contexts suggest a potential for stratified sampling – a well-known variance reduction technique wherein the input domain is divided into multiple disjoint sub-regions, each of which with discerning distributional behaviour of the outputs due to heteroscedasticity. The benefit of partitioning the data comes from the fact that the weighted average of conditional output variances is at most as large as the unconditional variance, that is,  $\mathbb{E}_{\mathbb{B}}[\text{Var}(A|B)] \leq \text{Var}(A)$  for any two random variables  $A$  and  $B$ . All stratified sampling research seeks to maximize this reduction in variance. Meanwhile, when used within optimization, one has to account for estimating the performance of many alternatives for the calibration parameter. Since the distribution of simulation outputs changes under each  $\theta$ , it is reasonable to consider that the optimal partitioning for each  $\theta$  should also vary. Hence, a priori partitioning of the input space (before starting the optimization algorithm) may be suboptimal. Therefore, the challenge of using stratified sampling within optimization can involve, in addition to choosing the sample size in each stratum, determining the best stratification structure for each  $\theta$  evaluated during optimization. With existing research addressing the former Jain et al. (2023), in this paper we focus on the latter and its integration within the optimization algorithm. We explore whether or not there is efficiency gains in choosing input sub-regions carefully for each iteration if it can be done at a low cost.

To that end, in a survey of stratified sampling techniques for estimation (Section 3), we investigate the risks in allocating budget to each stratum due to poor estimates of conditional variance of outputs, which can be particularly concerning if the strata boundaries keep changing during optimization. We discuss that, for dynamic strata, post-stratified sampling is a safe choice due to keeping the sampling distribution independent of the stratification structure. That is, we adopt independent sampling from the entire input space and then assign weights to each sample based on the placement of strata. We further leverage post-stratification variance as the metric when dynamically choosing the strata.

Next, we present two approaches to finding effective stratification structures (Section 4). The first method divides the data via binary trees (BT) for a greedy optimization of an objective function different from the standard trees. We propose an information gain metric to evaluate the reduction in variance

achieved after splitting a node and use that to choose the number of strata. The second method uses a linear relationship between some transformation of auxiliary or concomitant variables (ConV) and the outputs to approximate closed-form boundaries. Sometimes, rapidly computable conditional means of input (not simulated) data as concomitant variables can be used, reducing the small simulated sample risks on strata misspecification. We extend the closed-form derivations for variables to simulated data and propose ideas to choose the best concomitant variable and the number of strata in each iteration. Increased learning ability using all available data without needing new simulation runs and faster computation are the advantages of the second approach. Yet, it uses one concomitant variable at a time and hence may be less flexible than the first approach, which can stratify multiple variables jointly.

Lastly, we conduct a thorough numerical experiment (Section 5) with dynamically stratified adaptive sampling with BT and ConV and compare their performance with Bayesian optimization (BO), SA, and RS on Monte Carlo and discrete-event simulation toy examples. Further, implementation in a case study for the wind power simulation model calibration proves the effectiveness of the proposed methods in a real-world application and provides insights on their sensitivity and consistency. Comparisons are summarized with emphasis on remaining gaps and open questions for future research (Section 6).

## 2. Mathematical background and contributions

In this section, we define the data-driven calibration problem as a simulation optimization. We review its computational challenges, to remedy which we focus on a particular class of optimization algorithms that uses adaptive sampling within trust regions. We then list our contributions and gained insights that render this algorithm more successful for calibration.

### 2.1. Problem formulation and standing assumptions

Consider the random instances of data  $(X, Y)$  being generated from an underlying joint probability distribution and let  $h(\theta, X)$  be the simulated random output corresponding to the vector pair  $(X, Y)$ . Then, defining the loss function  $\ell(h(\theta, X), Y)$  as a measure of discrepancy between  $h(\theta, X)$  and  $Y$ , the objective function becomes

$$\min_{\theta \in \Theta} f(\theta) := \mathbb{E}_{X, Y}[\ell(h(\theta, X), Y)]. \quad (2)$$

The function  $f(\theta)$  is non-negative, and we assume that it is nonconvex (due to nonconvexity of  $h(\cdot, X)$ ) but

continuously differentiable with Lipschitz continuous gradients, i.e., there exists a constant  $L < \infty$  such that  $\|\partial f(\theta_1) - \partial f(\theta_2)\| \leq L \|\theta_1 - \theta_2\|$  for any  $\theta_1, \theta_2 \in \Theta$ . For the remainder of this paper, we simplify the random objective function notation with  $F(\theta, (X, Y)) := \ell(h(\theta, X), Y)$ . The actual joint probability distribution in (2) is unknown. Thus, we estimate the expected value at a particular  $\theta$  in (2) via Sample Average Approximation (SAA) using a random sample set  $\mathcal{S}$  of size  $N$  from the available dataset, i.e.,

$$\hat{f}(\theta, N) := \frac{1}{N} \sum_{(x_j, y_j) \in \mathcal{S}} F(\theta, (x_j, y_j)).$$

In other words, we consider each evaluation of the objective function for a single data point as one simulation replication, and the total evaluations throughout the optimization are accounted for as the computational budget needed to reach an optimal solution. Under the Big Data context, i.e., large  $n$ , the selection of the points will be considered in an identically distributed and independent (i.i.d.) fashion. Most simulation models  $h(\theta, \cdot)$  are too complex to have direct access to their derivatives with respect to  $\theta$ . Although direct gradients can be computable via techniques such as infinitesimal perturbation analysis (Suri, 1987), in most instances, that requires additional programming and analysis that may not be a feasible option in many applications. Hence, we consider the simulation model as a complete black-box and assume that  $\partial F(\theta, \cdot)$  are unavailable, which makes this problem a *derivative-free optimization* (DFO) (Conn et al., 2009).

### 2.2. Efficiency challenges for derivative-free simulation optimization

DFO problems are much harder to solve due to the needed extra effort to approximate gradients through derivative-free methods. Therefore, the main challenge is: can we obtain good solutions with an optimization algorithm in this setting given a fixed computational budget? The answer to this question involves the trade-off between exploration and exploitation that, while primarily known in Bayesian optimization, is a general challenge with optimization algorithms evaluated in finite time. In a deterministic viewpoint, exploitation refers to evaluating the objective function value at multiple  $\theta$ 's within a sub-region to track it locally. Expending a lot of budget for exploitation leaves less budget for the algorithm to explore other regions of the search space. Using the objective function's structure to determine the number of  $\theta$ 's is not a viable option in DFO. Instead, DFO solvers expend budget to approximate the gradients with interpolation or



finite differencing, among other methods. In the stochastic DFO, if the simulation outputs are very noisy, the approximated gradient can be inaccurate, and these methods can struggle to reach good solutions. Hence, exploitation in a stochastic setting involves both the number of  $\theta$ 's visited to approximate the gradient and estimating the objective function at each of those  $\theta$ 's.

Trust-region (TR) methods are increasingly known to be effective for non-convex DFO problems compared to line search or stochastic gradient methods due to their strict control of step size (tuned automatically throughout the search) and their implicit use of curvature by constructing a quadratic local model (Y. X. Yuan, 2015). The performance of TR depends on the quality of these local models as their high-quality can consistently identify good steps and progress per iteration.

However, the challenge with building high-quality models stems from the estimation error, which with  $N$  Monte Carlo samples are inversely proportional to  $\sqrt{N}$ . Thus, given a fixed budget, having reliable estimates that can help build better local models (exploitation) comes at the cost of losing the budget to take more steps (exploration). An efficient algorithm appropriately determines the sample size at each point within the local sub-region while keeping enough budget for exploration. A successful strategy for this trade-off *adapts* the choice of  $\mathcal{S}(\theta)$  as a function of  $\theta$  to the variability of  $F(\theta, \cdot)$  and to the precision stipulated for convergence, i.e., more accurate models and function estimates as the algorithm nears the optimal region (Byrd et al., 2012). Recently, a TR-based algorithm that incorporates adaptive sampling for the DFO problems, and hence is appropriate for data-driven calibration, has been developed, called ASTRO-DF (Adaptive Sampling Trust-Region Optimization – Derivative-Free) (Shashaani et al., 2016, 2018). In this paper, we will use this algorithm as an instance of adaptive sampling solvers and investigate on how to tailor it for data-driven calibration. As discussed in Section 1, our goal will be to integrate dynamic stratification within this solver.

### 2.3. Adaptive sampling trust-region optimization – derivative-free

ASTRO-DF is an almost surely convergent simulation optimization solver for nonconvex problems that builds a local model in each iteration via interpolation and decides the number of simulation runs (samples) based on a proxy for optimality gap. The adaptive sample size guarantees the fastest proven sample complexity of  $\mathcal{O}(\epsilon^{-4} \log \epsilon^{-1})$  to reach  $\epsilon$ -optimality (Ha et al., 2024). Had the direct gradient observations

been available, the lower-complexity solver (ASTRO – the derivative-based version of ASTRO-DF) could be used (Vasquez et al., 2019).

To better understand the sampling mechanism, let us briefly review the TR methods. For  $\theta_k$  as the iterate (incumbent solution) at iteration  $k$ , a TR is defined as a closed ball around  $\theta_k$ ,  $\mathcal{B}_k = \{\theta : \|\theta - \theta_k\|_2 \leq \Delta_k\}$ , where  $\Delta_k$  is the TR radius. A local model  $M_k(\cdot)$  is then fitted to estimated objective function values at multiple  $\theta$ 's inside  $\mathcal{B}_k$ . This model suggests a candidate for the next incumbent,  $\tilde{\theta}_{k+1}$  by predicting where the function will be minimized within  $\mathcal{B}_k$ . The reduction in the estimated objective function value at  $\tilde{\theta}_{k+1}$  is then compared to the corresponding reduction in the model. If sufficient reduction in the objective function value is achieved at  $\tilde{\theta}_{k+1}$ , then the candidate solution is accepted, i.e.,  $\theta_{k+1} = \tilde{\theta}_{k+1}$ , and the TR expands. If rejected, then  $\theta_{k+1} = \theta_k$ , the TR radius shrinks, and a new model is constructed in a smaller neighbourhood around  $\theta_k$ . For a complete listing of the new variant of this algorithm that uses the new proposed approaches for dynamic stratification, see Section 3.4.

In a (deterministic) DFO setting, the TR model gradient (in Euclidean norm) is maintained in lock-step with the TR radius, i.e.,  $\|M_k(\theta_k)\| = \mathcal{O}(\Delta_k)$  (Conn et al., 2009). Then, proving that  $\lim_{k \rightarrow \infty} \Delta_k = 0$  guarantees that the model gradient converges to 0. Handling the stochasticity comes in when one also needs to maintain a lock-step between the model gradient and the true function gradient. This is, in effect, what a high-quality model needs to accomplish in every iteration. ASTRO-DF deals with this challenge by choosing the optimal (i.e., most efficient) sample size at each visited  $\theta$ . Since the sequence  $\{\Delta_k\}$  shrinks as the algorithm nears optimality, ASTRO-DF uses the fourth power (appropriately selected to maintain model quality) of the TR radius as the acceptable upper bound for the standard error at each visited  $\theta$ . The sample size is, therefore, a *stopping time* of the form

$$N_k = \min \left\{ n \geq \lambda_k : \sqrt{\widehat{\text{Var}}(\hat{f}(\theta_k, n))} \leq \kappa \frac{\Delta_k^2}{\sqrt{\lambda_k}} \right\}, \quad (3)$$

since with every added sample, the LHS (standard error estimate) changes and eventually reduces while the RHS (slightly deflated optimality gap proxy) remains unchanged.

In (3),  $\lambda_k$  is a deterministic sequence that increases logarithmically with  $k$ , and  $\kappa$  is a positive constant. The deflation ensures that the acceptable standard error threshold is stricter in the later iterations and is essential for proving almost sure model quality guarantees (and hence algorithm convergence) (Ha & Shashaani, 2023). Another role of  $\lambda_k$  is lower bounding the sample size so that even under early stopping due

to a poor estimate of the standard error,  $N_k$  increases at least logarithmically to increase estimation accuracy. The algorithm first runs  $\lambda_k$  i.i.d. replications to obtain  $\widehat{\text{Var}}(\hat{f}(\theta_k, \lambda_k))$  and, if needed, adds one sample at a time. As a result, the adaptive sample size is small during the initial iterations when the optimality gap is large and increases in the later iterations when the algorithm appears to have neared optimality.

Choosing  $N_k$  provides theoretical guarantees for efficiency, but since there is no upper limit to the stopping time, it can practically be very large due to high noise in  $F(\theta_k, \cdot)$ . High level of noise is notoriously present in data-driven calibration causing extremely large sample sizes, which is undesirable under a finite budget setting. Enhancing the algorithm with a variance reduction technique such as stratified sampling (leveraging the conditional behaviour of the outputs in input sub-regions) can help avoid such larger sample sizes. However, a seamless incorporation of stratified sampling with adaptive sampling is challenging as which stratum to sample from in each recursion of the stopping time induces more uncertainty to the algorithm. There are also risks and opportunities in selecting the strata themselves appropriately in each iteration.

### 3. Stratified sampling for optimization

Stratified sampling groups similar data into strata such that the output within each group is similar and between any two groups is different. This helps learn the heterogeneity in the data for estimation, which leads to variance reduction (Ross, 2013). Intuitively, to efficiently allocate overall samples to each stratum, more points should be sampled from a stratum with higher variance. This efficient allocation of the computational budget can reduce the variance of the estimators and expedite the optimization. The impact of stratified sampling on the optimization routine is influenced by (i) the allocation scheme used to determine the sample size of each stratum and (ii) the stratification structure.

**The allocation scheme** depends on what sampling strategy is utilized. *Proportional* allocation sets the sample size of a stratum based on the probability of picking a point from that stratum. *Optimal* allocation depends on the above probability and the output variance in that stratum (Neyman, 1934). An inaccurate estimate of these two values can reduce the effectiveness of stratified sampling and produce worse estimates of the performance. Thus, many studies in the literature have averted to explore different ways to solve the problem of sample size allocation. Mathematical techniques like convex programming (Huddleston et al., 1970), branch and bound methods (Bretthauer et al., 1999), and delta

method (Glynn & Zheng, 2021) have been proposed to determine the optimal allocation. Since optimal allocation can be erratic if the variance cannot be estimated accurately, a hybrid allocation scheme that switches between proportional and optimal allocation as more insights are gained by running simulation can also be used (Pettersson & Krumscheid, 2021). Another common method is an adaptive optimal allocation that minimizes the variance within each stratum (Etoré & Jourdain, 2010; Kawai, 2010). All these studies focus on applying stratified sampling for simulations or statistical inference. Within the optimization framework, the optimal allocation has been implemented via batching (Chen et al., 2018; Hassan et al., 2006; Zhao & Zhang, 2014). One of the drawbacks of batching is that the sample sizes can be larger than specified by adaptive sampling and may result in inefficient budget utilization.

**The stratification structure** can be determined by partitioning the data based on an input variable such that output data in each stratum exhibits similar probabilistic characteristics. Consequently, we can assume a separate conditional distribution for the outputs in each stratum. This problem has been widely analysed for one-time stratification (for inference) using heuristics like clustering (Farias et al., 2020; Tipton, 2013; Zhao & Zhang, 2014), genetic algorithms (Keskintürk & Er, 2007), binary trees (Jain et al., 2021, 2022), etc. A drawback of these heuristics and greedy search methods is their reliance on the data used to build the structure, being susceptible to poor performance if the data is noisy or insufficient. Another approach is to use a theoretically derived closed-form solution to divide the data via concomitant variables (Dalenius, 1950). Concomitant variables are traditionally simulated input data with known mean and variance. If the concomitant variables are correlated to the outputs, the strata boundaries that minimize the variance can be determined by solving implicit equations derived given their conditional distributions (Dalenius & Gurney, 1951; Taga, 1967). However, the closed-form boundaries' equations are only solvable if the concomitant variables follow a well-known probability distribution (Sethi, 1963; Singh & Sukhatme, 1969). Otherwise, iterative fixed-point methods (Cochran, 1977; Sethi, 1963), convex optimization (Brito et al., 2010; de Moura Brito et al., 2017), and dynamic programming (Khan et al., 2008) have been used to solve these equations. Using these approaches for optimization can be computationally expensive given that they will be invoked at every iteration of the algorithm. In addition, they are only applicable when the number of strata and the stratification variable are known a priori, both of which may also vary from one iterate to another during optimization.

### 3.1. Notations and definitions

Consider a stratification structure  $\mathcal{I}_k$ , which divides the input space into  $Z_k$  disjoint sets  $(\mathcal{X}_{k,1}, \mathcal{X}_{k,2}, \dots, \mathcal{X}_{k,Z_k})$  such that  $\mathcal{X} := \cup_{z=1, \dots, Z_k} \mathcal{X}_{k,z}$  is the whole input space. For a sample  $\mathcal{S}_k := \mathcal{S}(\theta_k)$  of size  $N_k$  formed by subsamples  $\mathcal{S}_{k,z} = \{(x_j, y_j) \in \mathcal{S}_k : x_j \in \mathcal{X}_{k,z}\}$  of size  $N_{k,z}$  in each stratum (i.e.,  $N_k = \sum_{z=1}^{Z_k} N_{k,z}$ ), the estimated stratified sampling mean is

$$\hat{f}_{\text{strat}}(\theta_k, N_k | \mathcal{I}_k) = \sum_{z=1}^{Z_k} p_{k,z} \hat{f}_z(\theta_k), \quad (4)$$

where  $p_{k,z} = \mathbb{E}[1(X \in \mathcal{X}_{k,z})]$  is the probability of drawing a sample whose input lies in stratum  $z$ , and  $\hat{f}_z(\theta_k)$  is the sample average in stratum  $z$ , i.e.,

$$\hat{f}_z(\theta_k) = \frac{1}{N_{k,z}} \sum_{(x_j, y_j) \in \mathcal{S}_{k,z}} F(\theta_k, (x_j, y_j)).$$

Throughout this article, we assume  $p_{k,z} = \frac{|\mathcal{X}_{k,z}|}{|\mathcal{X}|}$  given the big data setting. The variance of the stratified sampling estimator is

$$\text{Var}(\hat{f}_{\text{strat}}(\theta_k, N_k | \mathcal{I}_k)) = \sum_{z=1}^{Z_k} \frac{p_{k,z}^2 \sigma_{k,z}^2}{N_{k,z}},$$

where  $\sigma_{k,z}^2 := \mathbb{E}[(F(\theta_k, (X, Y)) - f_z(\theta_k))^2 | X \in \mathcal{X}_{k,z}]$  is the variance of outputs in stratum  $z$  with  $f_z(\theta_k) := \mathbb{E}[F(\theta_k, (X, Y)) | X \in \mathcal{X}_{k,z}]$  as its mean. The estimated mean and variance of the stratified sampling estimator depend on the stratification structure  $\mathcal{I}_k$  and the set of sampled points  $\mathcal{S}_k$ . The reduction in the variance of the stratified sampling estimator given  $\mathcal{I}_k$  depends on how the samples are allocated to each stratum.

### 3.2. Review: Proportional vs. optimal allocation

For ease of exposure, let  $N_k$  be a deterministically growing sample size instead of a stopping-time sample size chosen adaptively for the remainder of this section. In proportional allocation,  $N_{k,z} = p_{k,z} N_k$  whereas in optimal (or Neyman) allocation  $N_{k,z} = w_{k,z} N_k$ , with weights computed as

$$w_{k,z} = \frac{p_{k,z} \sigma_{k,z}}{\sum_{z'=1}^{Z_k} p_{k,z'} \sigma_{k,z'}}.$$

Optimal allocation results in the lowest variance if  $\sigma_{k,z}$ 's are known for all  $z$  with

$$\text{Var}(\hat{f}_{\text{os}}(\theta_k, N_k | \mathcal{I}_k)) = \frac{1}{N_k} \left( \sum_{z=1}^{Z_k} p_{k,z} \sigma_{k,z} \right)^2 \text{ and}$$

$$\text{Var}(\hat{f}_{\text{ps}}(\theta_k, N_k | \mathcal{I}_k)) = \frac{1}{N_k} \sum_{z=1}^{Z_k} p_{k,z} \sigma_{k,z}^2,$$

for the optimal and proportional estimator, respectively. However, since estimates of the conditional variance  $\sigma_{k,z}$  in each stratum, i.e.,

$$\hat{\sigma}_{k,z}^2 := \frac{1}{N_{k,z} - 1} \sum_{(x_j, y_j) \in \mathcal{S}_{k,z}} (F(\theta_k, (x_j, y_j)) - \hat{f}_z(\theta_k))^2, \quad (5)$$

ought to be used instead, optimal allocation is subject to risks due to inaccurate  $\hat{\sigma}_{k,z}$ 's, which is more prominent in the early iterations. On the other hand, when using proportional allocation, the sample size of stratum  $z$  depends only on  $p_{k,z}$ , which can be more rapidly estimated using all the available input data. The maximum reduction in variance with optimal allocation is mostly effective in the later iterations for another reason too. Let  $N'_{k,z}$  be the theoretical optimal sample size of stratum  $z$ ,  $\text{Var}_{\text{opt}}$  be the variance of optimal allocation,  $\hat{N}_{k,z}$  be the estimated sample size of stratum  $z$  with optimal allocation and  $\text{Var}_{\text{est}}$  be the corresponding variance without stratification such that  $N_k = \sum_z N'_{k,z} = \sum_z \hat{N}_{k,z}$ . Increased variance due to under-or over-estimating the optimal allocation sample sizes can be characterized as

$$\text{Var}_{\text{est}} \geq \text{Var}_{\text{opt}} \left( 1 + \frac{1}{N_k} \sum_{z=1}^{Z_k} \frac{(\hat{N}_{k,z} - N'_{k,z})^2}{\hat{N}_{k,z}} \right),$$

which is more significant when  $N_k$  is small (Cochran, 1977). For early iterations of optimization with small  $N_k$  (where the algorithm trajectory is most vulnerable for progress), using proportional allocation is almost as good as the reduction with optimal allocation (Pettersson & Krumscheid, 2021). In fact, one can be more confident to obtain *any* reduction in the variance with proportional allocation than with the optimal allocation (Asmussen & Glynn, 2007, p. 151). In summary, using proportional allocation for the purpose of optimization reduces the run-to-run variability of the algorithm.

### 3.3. Allocation schemes with adaptive sampling

Section 3.2 discusses batch-based allocation. However, in implementation of stratified sampling in combination with adaptive sampling rules such as those in ASTRO-DF, we need to allocate each sample sequentially, one at a time. First, (3) is replaced by the standard error on the LHS with the standard error of the stratified sampling estimator. Next, if the standard error exceeds the optimality gap, it is necessary to determine from which stratum should an additional point be sampled. A selective randomized method proposes selecting a random stratum to sample from using the probability mass function

$$\begin{aligned} \pi_{k,z}(n) &= \Pr\{\text{selecting stratum } z \text{ after } n \text{ samples}\} \\ &= \frac{v_{k,z}(n) \mathbf{1}\{v_{k,z}(n) > \frac{n_z}{n}\}}{\sum_{z'=1}^{Z_k} v_{k,z'}(n) \mathbf{1}\{v_{k,z'}(n) > \frac{n_{z'}}{n}\}}, \end{aligned} \quad (6)$$

where  $n_z$  is the current number of  $n$  samples that belong to stratum  $z$  and

$$v_{k,z}(n) = \begin{cases} \hat{w}_{k,z}(n), & \text{for optimal allocation} \\ p_{k,z}, & \text{for proportional allocation,} \end{cases}$$

with  $\hat{w}_{k,z}$  as the estimate of  $w_{k,z}$  using  $n_z$  samples. The expected sample size in stratum  $z$  for a fixed iterate  $\theta_k$  conditional on the stopping time is

$$\begin{aligned} [N_{k,z}|N_k = n_k] &= \mathbb{E} \left[ \begin{array}{l} \text{zselections in pilot runs} + \\ \sum_{n=\lambda_k+1}^{n_k} \mathbf{1} \times \mathbb{P}\{\text{zselected on } n\text{th run}\} \end{array} \right] \\ &\approx \mathbb{E}[v_{k,z}(\lambda_k)]\lambda_k + \sum_{n=\lambda_k+1}^{n_k} \mathbb{E}[\pi_{k,z}(n)]. \end{aligned}$$

The approximation reveals the complication with analysing the sample size of each stratum. Although  $v_{k,z}$  has less variability in proportional allocation than in optimal allocation, even proportional allocation can result in instability simply because  $(N_{k,1}, N_{k,2}, \dots, N_{k,Z_k})$  is a multinomial random vector with each mode's probability changing sequentially with every new sample added following (6). In summary, when using adaptive sampling, both allocation schemes are subject to the changing sampling distribution with every added sample. It can cause unstable updating of  $\theta_k$  in the optimization process.

### 3.4. Post-stratification for stratified adaptive sampling with changing strata in optimization

Stratified adaptive sampling has been explored using optimal allocation and its extensions for stochastic gradient methods (Espath et al., 2021; B. Liu et al., 2022) and Nelder Mead (Aguiar et al., 2022) Jain et al., 2021, 2022 examined how robust is the implementation of stratified adaptive sampling for TR methods. However, stratified sampling requires the stratification structure to be known a priori to sample points from each stratum independently. Consequently, most existing studies use a constant stratification structure, i.e.,  $\mathcal{I}_k = \mathcal{I}$  for all  $k$ , to maintain a consistent sampling framework throughout the search. Even with the fixed structure, stratified sampling with optimal (or proportional) allocation faces increased stochasticity when the sampling distribution changes in the optimization process, as discussed in Section 3.3. To overcome these vulnerabilities, one can use post-stratification, which first samples randomly from the entire population. Then the estimation follows similar to stratified

sampling with proportional allocation using  $N_{k,z}$ , the number of sampled points that are within each stratum. Central limit theorems for proportional allocation apply to post-stratification (Asmussen & Glynn, 2007), and its finite-time performance in queuing simulations has been on par with variance reduction obtained using control variates (Wilson et al., 1984).

The post-stratified sampling estimator  $\hat{f}_{\text{post}}(\theta_k, N_k|\mathcal{I}_k)$  is evaluated via (4) to obtain each  $\hat{f}_z(\theta_k)$ ; its variance is then exactly computed (Cochran, 1977) as

$$\begin{aligned} \text{Var}(\hat{f}_{\text{post}}(\theta_k, N_k|\mathcal{I}_k)) &= \frac{1}{N_k} \sum_{z=1}^{Z_k} p_{k,z} \sigma_{k,z}^2 \\ &+ \frac{1}{N_k^2} \sum_{z=1}^{Z_k} (1 - p_{k,z}) \sigma_{k,z}^2 \\ &+ \mathcal{O}\left(\frac{1}{N_k^3}\right). \end{aligned} \quad (7)$$

The first term in (7) is the variance of the proportional allocation, and the second term is the increase in variance because the post-stratification does not account for the stratification structure. Post-stratification is not an allocation scheme as the allocation happens automatically. Importantly, this reduces variability since  $(N_{k,1}, N_{k,2}, \dots, N_{k,Z_k})$  is now a multinomial random vector with each mode's probability determined by  $p_{k,z}$ , and hence fixed. From (5), the reduced variability manifests in more stable estimates for the conditional variance in each stratum. In other words, under adaptive sample sizes,  $\hat{\sigma}_{k,z}$  obtained via post-stratification is a better estimate for  $\sigma_{k,z}$  than that obtained via the proportional allocation.

When we want to let the stratification structure  $\mathcal{I}_k$  change with the decision variable  $\theta_k$ , post-stratification will again be more appropriate since it does not necessitate a priori knowledge of the stratification structure when drawing samples. Therefore, we can construct the stratification structure using the pilot simulations and estimating the post-stratified variance estimator (7). This means  $\lambda_k$  in (3) needs to be significantly larger than in the standard ASTRO-DF to start, but ultimately, it can save a budget for exploration later in the search. In Section 4, we will present two approaches for constructing the strata from the learning yielded by the  $\lambda_k$  pilot samples.

Once the stratification structure is ready, we let the automatic allocation of samples to each stratum be executed, while we only take i.i.d. samples from the input space and leverage the adaptive sampling (to decide when to stop taking more samples) with less effort. If the standard error of the post-stratified estimator is more than the RHS in (3), one point is randomly sampled from the entire data that will lie in one of the strata depending on the stratification structure. Then, the estimates are updated, and the



adaptive sampling rule is examined again, and this process repeats until (3) is satisfied. Algorithm 1 outlines the implementation of ASTRO-DF for a given stratification structure  $\mathcal{I}_k$ , and the details of post-stratified adaptive sampling are summarized in Algorithm 2. The output of Algorithm 2 will be denoted depending on whether the input is  $\theta_k$  (iterate),  $\theta_k^i$  (interpolation points), or  $\tilde{\theta}_{k+1}$  (candidate solution).

---

**Algorithm 1** ASTRO-DF with Dynamic Post-Stratification
 

---

1: **Input:** Initial solution  $\theta_0$  and TR radius  $\Delta_0$ , maximum budget  $T$ , and success threshold  $\eta > 0$ .  
 2: **initialization:** Set calls = 0 and iteration  $k = 0$ .  
 3: **while** calls  $< T$  **do**  
 4: Estimate  $\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k)$  and  $\widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))$  via Alg. 2.  
 5: Set calls = calls +  $N_k$ .  
 6: Select  $2d$  points using the coordinate basis in  $\mathcal{B}_k$ , i.e.,  $\{\theta_k \pm \Delta_k e_i\}_{i=1}^d$ .  
 7: Estimate interpolation points' function value with  $N_k^i$  samples via Alg. 2.  
 8: Set calls = calls +  $\sum_{i=1}^{2d} N_k^i$ .  
 9: Construct model  $M_k(\theta)$  by interpolation and find  $\tilde{\theta}_{k+1}$ , its minimizer in  $\mathcal{B}_k$ .  
 10: Estimate  $\hat{f}_{\text{post}}(\tilde{\theta}_{k+1}, \tilde{N}_{k+1} | \tilde{\mathcal{I}}_{k+1})$  and  $\widehat{\text{Var}}(\hat{f}_{\text{post}}(\tilde{\theta}_{k+1}, \tilde{N}_{k+1} | \tilde{\mathcal{I}}_{k+1}))$  via Alg. 2.  
 11: Set calls = calls +  $\tilde{N}_{k+1}$ .  
 12: Compute the success ratio  $\hat{\rho}_k = \frac{\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k) - \hat{f}_{\text{post}}(\tilde{\theta}_{k+1}, \tilde{N}_{k+1} | \tilde{\mathcal{I}}_{k+1})}{M_k(\theta_k) - M_k(\tilde{\theta}_{k+1})}$ .  
 13: **if**  $\hat{\rho}_k > \eta$  **then**  
 14: Set  $\theta_{k+1} = \tilde{\theta}_{k+1}$  and  $\Delta_{k+1} > \Delta_k$ .  
 15: **else**  
 16: Set  $\theta_{k+1} = \theta_k$  and  $\Delta_{k+1} < \Delta_k$ .  
 17: **end if**  
 18: Set  $k = k + 1$ .  
 19: **end while**  
 20: **output:** Terminal solution  $\theta_k$  and terminal performance  $f(\theta_k) \approx \mathbb{E}[F(\theta_k, (X, Y))]$ .

---



---

**Algorithm 2** Post-Stratified Adaptive Sampling
 

---

1: **Input:** Available dataset  $\mathcal{X}$ , iterate  $\theta_k$ , TR radius  $\Delta_k$ , minimum sample size  $\lambda_0$ , and constant  $\kappa > 0$ .  
 2: Compute  $\lambda_k = \lambda_0 (\log k)^{1.5}$ .  
 3: Run  $\lambda_k$  i.i.d. simulations to obtain  $F(\theta_k, (x_j, y_j)) \quad j = 1, 2, \dots, \lambda_k$ . Set  $N_k = \lambda_k$ .  
 4: Generate a stratification structure  $\mathcal{I}_k$  with  $Z_k$  strata via Alg. 3 or Alg. 4.  
 5: Compute  $\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k)$ ,  $\widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))$  with  $N_k = \sum_{z=1}^{Z_k} N_{k,z}$  using  $\mathcal{I}_k$ .  
 6: **while**  $\sqrt{\widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))} > \kappa \frac{\Delta_k}{\sqrt{\lambda_k}}$  **do**  
 7: Take an i.i.d. sample and identify stratum  $z$  that it belongs to based on  $\mathcal{I}_k$ .  
 8: Set  $N_{k,z} = N_{k,z} + 1$  and  $N_k = N_k + 1$ .  
 9: Update  $\hat{f}_z(\theta_k)$ ,  $\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k)$ , and  $\widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))$ .  
 10: **end while**  
 11: **output:** Sample size  $N_k$  and estimates  $\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k)$ ,  $\widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))$ .

---

## 4. Dynamic construction of strata

An important aspect of Algorithm 1 is determining the stratification structure  $\mathcal{I}_k$ . Constructing a stratification structure involves determining three things:

- (i) number of strata,
- (ii) stratification variable (when there are multiple input variables), and
- (iii) strata boundaries (split values).

A fixed structure can be built if physics-based (i)–(iii) are known a priori. In many practices, these will not be known, and while asynchronous or static stratification still has an advantage for variance reduction, not getting (i)–(iii) right in implementation may barely benefit the optimization if not slowing it down (Jain et al., 2022). The question is, can we do better than selecting the strata without consideration for the local conditional behaviour of the objective function? Especially for heteroscedastic problems, fixed strata may not be optimal at every iteration as the conditional output distribution can greatly vary at different  $\theta$ 's. Synchronous or dynamic stratification may thus be beneficial in building the optimal strata for each  $\theta$  that is evaluated during optimization if the computational cost of doing so is not too expensive.

Several methods have been proposed to build a dynamic structure via greedy search (Etoré et al., 2011; B. Liu et al., 2022; Pettersson & Krumscheid, 2021) that address (ii) and (iii) but assume there is always a fixed number of strata across iterations. More strata means more quantities that need to be estimated (mean and variance of each stratum). Obtaining maximal reduction in variance requires large samples in each stratum to accurately estimate their statistics. For too many strata, the budget utilization can thus be extremely high.

In this work, we propose two ways to determine the stratification structure by finding solutions to (i)–(iii) simultaneously such that

$$\mathcal{I}_k = \underset{\mathcal{I}}{\arg \min} \widehat{\text{Var}}\left(\hat{f}_{\text{post}}(\theta_k, N_k = \lambda_k | \mathcal{I})\right).$$

Strata Structure

We propose two methods for stratification: a greedy search with a new variant of binary trees enabling more complex strata, and a closed-form solution with only one stratification variable at a time but with applicable to either *real* or *simulated* data.

### 4.1. Stratification via binary trees

The first stratification method we present greedily divides the data with binary trees to minimize the estimated variance. The first step is to decide the stratification variable and the corresponding split point. Let  $z$  be the current leaf that is to be split and  $X^1, X^2, \dots, X^p$  be the possible options for stratification variable with  $\text{Rng}_{k,z}(X^t)$  as the set of all possible values that the  $t$ -th variable can take in leaf  $z$ . Splitting value will divide leaf  $z$  into two candidate leaves defined for sets of (all not samples of) input data  $\mathcal{X}_{k,z,l(t,x)} = \{X \in \mathcal{X}_{k,z} : X^t \leq x\}$  and  $\mathcal{X}_{k,z,r(t,x)} = \{X \in \mathcal{X}_{k,z} : X^t > x\}$  to the left and right of the splitting criteria, respectively. We denote the sample size and the estimated variance in the left and right candidate leaf after splitting as  $N_{k,z,l(t,x)}$ ,  $N_{k,z,r(t,x)}$ ,  $\hat{\sigma}_{k,z,l(t,x)}^2$ , and  $\hat{\sigma}_{k,z,r(t,x)}^2$ , respectively.

Then, the optimal stratification variable and the corresponding split point are determined by minimizing the variance of proportional allocation estimated after splitting, i.e.,

$$(X_{k,z}^*, x_{k,z}^*) = \arg \min_{t=1,2,\dots,p;x \in \text{Rng}_{k,z}(X^t)} \hat{\sigma}_{k,z,l(t,x)}^2 Q_{k,z,l(t,x)} + \hat{\sigma}_{k,z,r(t,x)}^2 Q_{k,z,r(t,x)}. \quad (8)$$

In (8), weights  $Q_{k,z,l(t,x)} = \frac{N_{k,z,l(t,x)}}{N_k} = \frac{|S_{k,z,l(t,x)}|}{|S_k|}$  and  $Q_{k,z,r(t,x)} = \frac{N_{k,z,r(t,x)}}{N_k} = \frac{|S_{k,z,r(t,x)}|}{|S_k|}$  are used from samples collected so far (i.e., with  $|S_k| = \lambda_k$  pilot samples) in place of probabilities  $p_{k,z,l(t,x)} = \frac{|X_{k,z,l(t,x)}|}{|\mathcal{X}|}$  and  $p_{k,z,r(t,x)} = \frac{|X_{k,z,r(t,x)}|}{|\mathcal{X}|}$  because finding the probabilities using the entire available data for every possible  $x \in \text{Rng}_{k,z}(X^t)$  can be computationally expensive.

We denote the optimal left and right splits from solving (8) with  $l^*$  and  $r^*$ . This optimization can be solved sequentially as the binary tree grows to provide an ultimate stratification via greedy search. The optimal stratification variable at each leaf can be different, and it depends on the current iterate  $\theta_k$  and the set of points sampled during that iteration. The split variables and values  $X_{k,z}^*, x_{k,z}^*$  determine strata boundaries, which can change with  $\theta_k$  and the sample size. Upon accepting a split, we update the indexing of the strata (leaves) by setting the index of the left candidate leaf as  $z$  and the index of the right candidate leaf as  $Z_k + 1$  and finally incrementing the total strata by 1, i.e.,  $Z_k = Z_k + 1$ . This means the old  $z$ -th leaf is now replaced with the left candidate leaf, and the right candidate leaf is added to the list of leaves and will not be considered for further splitting.

The next step is to determine when the algorithm should stop splitting the data or, in other words, the number of strata; each leaf of the tree will be a stratum at the end. One approach is to pre-select the maximum number of strata. Too many strata means many quantities to be estimated, and too few strata can mean we lose substantial variance reduction. The best choice for  $Z_k$  can differ from one iterate to another, and pre-selecting it is subjective. A natural solution can be cross-validation, whereby the prediction error that falls below a certain threshold would stop the stratification process. The issues with this approach are cutting the small sample of data used for constructing the tree even shorter to form cross-validation folds in addition to ad-hoc choice of the prediction error threshold, which could rely on problem-dependent hyperparameters. These issues can result in a suboptimal stratification structure, affecting the estimates and the optimization process. Instead, we present an approach that determines the necessity of splitting a leaf by assessing the extent of variance reduction achieved through the proposed split based on *information gain* (Quinlan, 1986).

Consider  $\mathcal{I}_k$  as the stratification structure built so far. Before splitting, we will collect a statistic from each leaf  $z = 1, 2, \dots, Z_k$ . Let  $\hat{\sigma}_k^2 = \widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, N_k | \mathcal{I}_k))$  be the estimated variance of the post-stratified estimator given the current structure  $\mathcal{I}_k$ , and the new variance of the post-stratified estimator if this leaf was selected for splitting using the criteria returned by (8) be evaluated as

$$\begin{aligned} \tilde{\sigma}_k^2(z) = & \hat{\sigma}_k^2 - \frac{1}{N_k} \left( (p_{k,z} + \frac{1-p_{k,z}}{N_k}) \hat{\sigma}_{k,z}^2 \right. \\ & \left. - (p_{k,z,l^*} + \frac{1-p_{k,z,l^*}}{N_k}) \hat{\sigma}_{k,z,l^*}^2 - (p_{k,z,r^*} + \frac{1-p_{k,z,r^*}}{N_k}) \hat{\sigma}_{k,z,r^*}^2 \right). \end{aligned} \quad (9)$$

Note, we use true probabilities in (9) for correct estimation of the reduced variance as they can be computed using the whole data once the split point is known. Since the strata are non-overlapping,  $p_{k,z} = p_{k,z,l^*} + p_{k,z,r^*}$  and  $N_{k,z} = N_{k,z,l^*} + N_{k,z,r^*}$ . We define

$$\delta_k(z) := \frac{\hat{\sigma}_k^2 - \tilde{\sigma}_k^2(z)}{\hat{\sigma}_k^2},$$

as the proportion of variance reduced when node  $z$  is split. Now, the question is whether this reduction in variance is enough to split the node. To answer that, we define

$$G_k(z) := -\delta_k(z) \ln(\delta_k(z)), \quad (10)$$

which can be viewed as the information gained by splitting the node  $z$  at  $\theta_k$ . Assuming that the split leads to a reduction in the estimated variance,  $\delta_k(z) \in (0, 1)$ , and hence the maximum theoretical value of  $G_k(z) = 1/e$ . This information gain value is computed for each leaf. Let  $G_k^{\text{prev}}$  be the gain from the most recent accepted split. Then, the leaf selected for splitting is the one that maximizes  $G_k(z)$  subject to providing a gain that is at least as good as the previous gain  $G_k^{\text{prev}}$ , i.e.,

$$\begin{aligned} & \max_{z=1,2,\dots,Z_k} G_k(z), \\ & \text{subject to: } G_k(z) > G_k^{\text{prev}}. \end{aligned} \quad (11)$$

The selected leaf is then indexed appropriately, and its gain updates the value of  $G_k^{\text{prev}}$ . If (11) has no feasible solutions, the splitting stops, and the tree and stratification structure is finalized. We also use another hyperparameter ( $\tau$ ) common for binary trees that removes leaves with less than a certain number of data points sampled as shown in Algorithm 3. Generally,  $G_k(z)$  is initially small, then it reaches the maximum value after the first few splits and starts reducing after that. The algorithm stops when  $G_k(z)$  is close to the maximum value because finding a split that results in more gain is difficult after that. Note,

another advantage of using the information gain strategy for splitting is that we can select which of the leaves provides the best split rather than splitting leaves in the order of their indexing.

---

**Algorithm 3** Determining Strata (multi-D) using Binary Trees at Iteration  $k$ 


---

1: **Input:** Current iterate  $\theta_k$ , minimum leaf size threshold  $\tau$ , and loss values computed at  $(x_j, y_j) \in \mathcal{S}_k$  where  $|\mathcal{S}_k| = \lambda_k$ .  
2: **Initialization:**  
3: Compute the first split by solving (8) to get  $\mathcal{X}_{k,1,r^*}$ ,  $\mathcal{X}_{k,1,r^*}$  and compute  $G_k(1)$ .  
4: Set  $G_k^{\text{prev}} = G_k(1)$ ,  $\mathcal{X}_{k,1} = \mathcal{X}_{k,1,r^*}$  and  $\mathcal{X}_{k,2} = \mathcal{X}_{k,1,r^*}$ .  
5: Set  $\mathcal{I}_k = \{\mathcal{X}_{k,1}, \mathcal{X}_{k,2}\}$ ,  $\mathcal{S}_{k,1} = \mathcal{S}_{k,1,r^*}$ , and  $\mathcal{S}_{k,2} = \mathcal{S}_{k,1,r^*}$ .  
6: Set  $Z_k = 2$  and update  $\hat{\sigma}_k^2$  following (5).  
7: **while true do**  
8: **for**  $z \in \{1, 2, \dots, Z_k : |\mathcal{S}_{k,z}| > 2\tau\}$  **do**  
9: Compute optimal split in  $z$ -th leaf by solving (8) to get  $\mathcal{X}_{k,z,r^*}$ ,  $\mathcal{X}_{k,z,r^*}$ .  
10: If  $\min\{|\mathcal{S}_{k,z,r^*}|, |\mathcal{S}_{k,z,r^*}|\} > \tau$ , compute  $G_k(z)$  via (10), else set  $G_k(z) = -\infty$ .  
11: **end for**  
12: **if** there is an acceptable split, i.e., optimization (11) has a solution **then**  
13: Set the leaf that solves (11) as  $z^{\text{split}}$  and remove  $\mathcal{X}_{k,z^{\text{split}}}$  from  $\mathcal{I}_k$ .  
14: Set  $G_k^{\text{prev}} = G_k(z^{\text{split}})$ ,  $\mathcal{X}_{k,z^{\text{split}}} = \mathcal{X}_{k,z^{\text{split}},r^*}$  and  $\mathcal{X}_{k,Z_k+1} = \mathcal{X}_{k,z^{\text{split}},r^*}$ .  
15: Set  $\mathcal{I}_k = \mathcal{I}_k \cup \{\mathcal{X}_{k,z^{\text{split}}}, \mathcal{X}_{k,Z_k+1}\}$ ,  $\mathcal{S}_{k,z^{\text{split}}} = \mathcal{S}_{k,z^{\text{split}},r^*}$ ,  $\mathcal{S}_{k,Z_k+1} = \mathcal{S}_{k,z^{\text{split}},r^*}$ .  
16: Set  $Z_k = Z_k + 1$  and update  $\hat{\sigma}_k^2$  following (5).  
17: **else**  
18: **break**  
19: **end if**  
20: **end while**  
21: **Output:** Stratification structure  $\mathcal{I}_k$  with  $Z_k$  many strata and samples  $\{\mathcal{S}_{k,z}\}_{z=1}^{Z_k}$ .

---

## 4.2. Stratification using concomitant variables

Trees can partition the input space with multiple variables simultaneously. Yet, their drawback is the greedy heuristic search that can have intense computation at every iteration and sensitivity to smaller subsets of  $\lambda_k$  pilot samples that estimate the leaf statistics. They are susceptible to producing less effective strata in the early iterations where the pilot run is small. They fall short of the attractive feature of asynchronous strata using large quantities of data (without running simulations).

We propose a second method to construct strata that will, to the extent possible, use large quantities of data while still enjoying dynamic stratification. To motivate this method, we review the two properties of an ideal stratification variable. First, since the basis of stratification is conditioning the simulation output, input variables are helpful given that their distributional behaviour can be inferred in each defined stratum without much burden. Second, a good stratification variable is one with a strong correlation (linear dependence) with the simulated output. In fact, it is possible to derive closed-form boundaries for input variables with known or partially known distributions that are linearly dependent on the stochastic objective function values. Variables that are auxiliary to the stochastic objective function value and are generated during a simulation run can hence be used in service of variance reduction; we term these variables, the concomitant variables.

Concomitant variables' use for constructing strata boundaries is reminiscent of control variates and exploiting their linear dependence with the random output of interest (Wilson et al., 1984). Jain and Shashaani (2023) use the derived closed-form boundaries using optimal allocation for a queuing problem whose total cost one wishes to minimize. Simulated data considered for this purpose either have known distributions (service times) or unknown distributions (waiting times). In both cases, the amount of data is limited because it is what the simulation runs will generate, but the waiting time appears to be a better concomitant variable given its more direct linear relationship with the total cost.

In this paper, we extend this viewpoint (by using proportional allocation instead of optimal allocation for stability) to the data-driven calibration with two new considerations: a) besides the simulation-generated data, we have a vast amount of real (not simulated) input data that can be used rapidly without running simulations to construct the strata; and b) to choose among real or simulated variables the most linearly dependent with the objective function, we include a number of their nonlinear transformations as potential candidates to serve as the concomitant variable. If the concomitant variable is chosen to be among the real input data, then it would provide the same boundaries given a number of strata for any visited  $\theta$ . But dynamic stratification will be due to the choice of the variable and the number of strata that can change from one iterate to another. We next describe different parts of the new approach, as laid out in Algorithm 4.

---

**Algorithm 4** Determining Strata (1-D) using Concomitant Variables at Iteration  $k$ 


---

1: **Input:** Current iterate  $\theta_k$ , maximum number of strata  $Z_{\max}$ , loss computed at  $(x_j, y_j) \in \mathcal{S}_k$  where  $|\mathcal{S}_k| = \lambda_k$ , and thresholds  $\varepsilon = 10^{-6}$  and  $\rho = 10^{-1}$ .  
2: **Initialization:** Collect candidate concomitant variables from linear/nonlinear transformation of real or simulated data  $\{C_k^1, C_k^2, \dots, C_k^r\}$ . Set  $Z = 1$ .  
3: **for**  $i = 1, 2, \dots, r$  **do**  
4: Fit a weighted regression model  $F(\theta_k, (X, Y))\alpha_k^i + \beta_k^i C_k^i + E_k^i$ .  
5: Estimate  $\text{Var}(E_k^i)$  and  $\text{Corr}(C_k^i, E_k^i)$ .  
6: **end for**  
7: Find  $C_k := C_k^{i^*}$  where  $i^* = \arg \min_{i=1,2,\dots,r} \{\widehat{\text{Var}}(E_k^i) : |\widehat{\text{Corr}}(C_k^i, E_k^i)| < \rho\}$ .  
8: **for**  $Z = 2, \dots, Z_{\max}$  **do**  
9: **if** distribution of  $C_k$  is known **then**  
10: Look up  $c_1 < c_2 < \dots < c_{Z-1}$  and set  $c_0 = -\infty$ ,  $c_Z = \infty$ .  
11: **else**  
12: Set  $c_0 = -\infty$ ,  $c_Z = \infty$  and  $c_1, c_2, \dots, c_{Z-1}$  as  $Z - 1$  quantiles of data.  
13: **repeat**  
14: Set  $c_z = c_z'$   $z = 1, 2, \dots, Z - 1$ .  
15: Estimate  $\mu_z = \mathbb{E}[C_k | c_{z-1} \leq C_k < c_z]$   $z = 1, 2, \dots, Z - 1$ .  
16: Set  $c_z' = (\mu_z + \mu_{z+1})/2$   $z = 1, 2, \dots, Z - 1$ .  
17: **until**  $\|(c_z')_{z=1}^Z - (c_z)_{z=1}^Z\| \leq \varepsilon$   
18: **end if**  
19: Set  $\mathcal{I}_{k,Z} = \{\mathcal{X}_{k,z}\}_{z=1}^Z$ , where  $\mathcal{X}_{k,z} = \{X : C_k(X) \in [c_z, c_{z+1})\}$ .  
20: **end for**  
21: Determine  $Z_k = \arg \min \widehat{\text{Var}}(\hat{f}_{\text{post}}(\theta_k, \lambda_k | \mathcal{I}_{k,Z}))$  via bootstrapping.  $Z \in [2, Z_{\max}]$   
22: **output:** concomitant variable  $C_k$  with  $Z_k$  many strata and structure  $\mathcal{I}_k = \mathcal{I}_{k,Z_k}$ .

---

**Boundaries on a concomitant variable:** Suppose  $C = C(X)$  is the concomitant variable used for

stratification – a linear/nonlinear transformation of a real variable in our dataset or a simulated variable generated alongside the simulated outputs of interest. Optimal stratification structure involves the boundaries

$$c_0 < c_1 < \dots < c_{Z_k},$$

that minimize the variance of the stratified sampling estimator to obtain the stratification structure  $\mathcal{I}_k$ .  $c_0$  and  $c_{Z_k}$  are the two extreme values for  $C$ , typically considered to be  $\pm\infty$ . Leveraging post-stratification, boundaries that minimize the variance can be derived using the following theorem:

**Theorem 4.1** (Dalenius and Gurney (1951)) Suppose the linear regression relation  $F = \alpha + \beta C + E$  holds with  $\mathbb{E}[E] = 0$ ,  $\text{Var}(E) = \sigma_E^2$ , and  $\text{Cov}(C, E) = 0$ . Suppose also that we have a total of  $n$  samples and want  $Z$  strata on  $C$ . Then, we can minimize the post-stratified estimator's variance to order  $n^{-1}$  by choosing the strata boundaries

$$\begin{aligned} c_z &= \frac{\mathbb{E}[C|c_{z-1} \leq C < c_z] + \mathbb{E}[C|c_z \leq C < c_{z+1}]}{2} \quad \forall z \\ &= 1, 2, \dots, Z - 1. \end{aligned} \quad (12)$$

While the closed-form equation (12) is recursive and appears complex, under known probability distribution of  $C$ ,  $c_z$  quantities can be exactly computed and are accessible in look-up tables for several distributions (Sethi, 1963). The standard normal case for  $C$  is relevant in simulation studies, providing good approximations for standardized variables that are aggregated statistics common in many discrete-event models. If the concomitant variable has an unknown distribution, we can solve for the optimal boundaries using a (relatively fast) convergent fixed-point iterative method (Burden & Faires, 1989); see Step 12-Step 17 in Algorithm 4. During these steps, if  $C$  is among the simulated data, then the conditional means are estimated with  $\lambda_k \ll n$  pilot runs. Big data has less leverage in this case, similar to the binary tree approach. If  $C$  is among the real data, its conditional means can be approximated with rapid population statistics to compute boundaries. In both cases, strata are exactly or approximately computed for a fixed number of strata.

**Choosing the concomitant variable.** During optimization, there may be different variables at different iterations that provide the most linear relationship with the outputs, i.e.,  $C_k$  such that

$$F(\theta_k, (X, Y)) = \alpha_k + \beta_k C_k + E_k,$$

where  $E_k$  is the stochastic residual satisfying the assumptions in Theorem 4.1. One can use transformations of the original real or simulated variables to find the desired linear relationship (either through

some descriptive data analysis or by using expert knowledge about the model). In this paper, we propose gathering a list of transformed variables as candidates for the concomitant variable and fitting a weighted least squares linear regression model for each candidate; see Step 3-Step 7 in Algorithm 4. The weighted least squares regression is preferred over ordinary least squares to account for the outliers, heterogeneous variance, and the erratic behaviour of the simulations (Holland & Welsch, 1977). We choose a  $C_k$  that yields the smallest *ratio of variance*  $\widehat{\text{Var}}(E_k)/\widehat{\text{Var}}(F_k)$ ; if the residual has a smaller variance than the response, then inference about the mean of the residual will be more precise than the same inference about the mean of the response (Smith, 1991). Importantly, making an inference about the population by training a regression model using the samples has the risk of incorrectly estimating the regression coefficients  $\alpha_k, \beta_k$ . The variance of the residuals is independent of the stratification structure yet affected by this erroneous estimation. We emphasize that these operations are relatively fast given the use of  $\lambda_k$  samples for fitting a whole bunch of regression lines. This is in contrast to the geometric number of operations in the binary tree to find the optimal stratification structure.

**Choosing the number of strata.** The last challenge is finding the number of strata. We decide the number of strata by determining  $Z_k$  to find the lowest variance. In other words, if  $\mathcal{I}_{k,Z}$  denotes the stratification structure with  $Z$  strata, we find

$$Z_k = \arg \min_{Z \in [2, Z_{\max}]} \widehat{\text{Var}}\left(\hat{f}_{\text{post}}(\theta_k, N_k = \lambda_k | \mathcal{I}_{k,Z})\right), \quad (13)$$

where  $Z_{\max}$  is the user-defined upper limit on the number of strata; Cochran (1977) proves that  $Z_{\max} = 6$  for regression models with  $R^2 < 0.95$ . To evaluate the variance given  $Z$  strata, we use  $n_{\text{boot}}$  bootstraps of  $\lambda_k$  simulated data to evaluate the variance for each  $Z$  and identify one that consistently yields the smallest variance.

## 5. Experimental results

We compare the proposed stratification methods (BT and ConV) to the corresponding trust-region-based method without stratification (NS) and a number of widely used global optimization methods, including Bayesian Optimization (Frazier, 2018), Simulated Annealing (Prudius & Andradóttir, 2012), and Random Search (Andradóttir, 2006). Our numerical analysis spans Monte Carlo examples with various variance structures (heterogeneity) and input dimensions, queuing simulations, and a data-driven calibration case



study with real data from an offshore wind farm. In all of these problems, the objective (loss) function is the mean squared error (MSE) as in ERM cases (1), quantifying the discrepancy of the simulated and observed data. By minimizing this MSE, we aim to calibrate the simulation model  $h(\theta, X)$  by finding the optimal parameter value  $\theta$ . Each solver is run 20 independent times (20 macroreplications), given a fixed computational budget, to obtain a distribution of the optimal calibrated parameter values, starting from the same initial point,  $\theta_0$ . In each macroreplication, common random numbers (CRN) are used across solvers to enable reproducibility and sharper comparison. For BO, a combination of RBF and white kernel is used with expected improvement as the acquisition function. The RBF kernel is effective for building surrogate models in BO because it can approximate any function given enough data and is infinitely differentiable, ensuring that the surrogate model is very smooth. The scale parameter of the RBF kernel is tuned by maximizing the log-marginal likelihood. For NS, the initial sample size is set to 80 to scale adaptively based on the variance of the estimates. In BT, we use the minimum leaf size threshold  $\tau = 5$  while building the trees and for ConV, the maximum number of strata used is  $Z_{\max} = 4$ .

For the trust-region methods, we make comparisons from these large-scale experiments by reporting intermediate recommended solutions at different budget points to track the optimization trajectory. Aligned with the SimOpt library platform (Eckman et al., 2023), we post-process these solutions; the objective function value at these intermediate solutions is estimated using a validation set that is 30% of the total data sampled independent from the modeling set that generates the optimization trajectory. The post-estimated objective function values for each macroreplication of each solver are then aggregated to obtain the mean and 95% confidence intervals (CI) to obtain progress curves per expended budget.

We first present the results for some numerical examples in Section 5.1, followed by the results for queuing calibration in Section 5.2. In Section 5.3, we present the wind case study and present an in-depth analysis of the two stratification approaches. Finally, in Section 5.4, we discuss the scenarios in which each proposed method may perform better than the others.

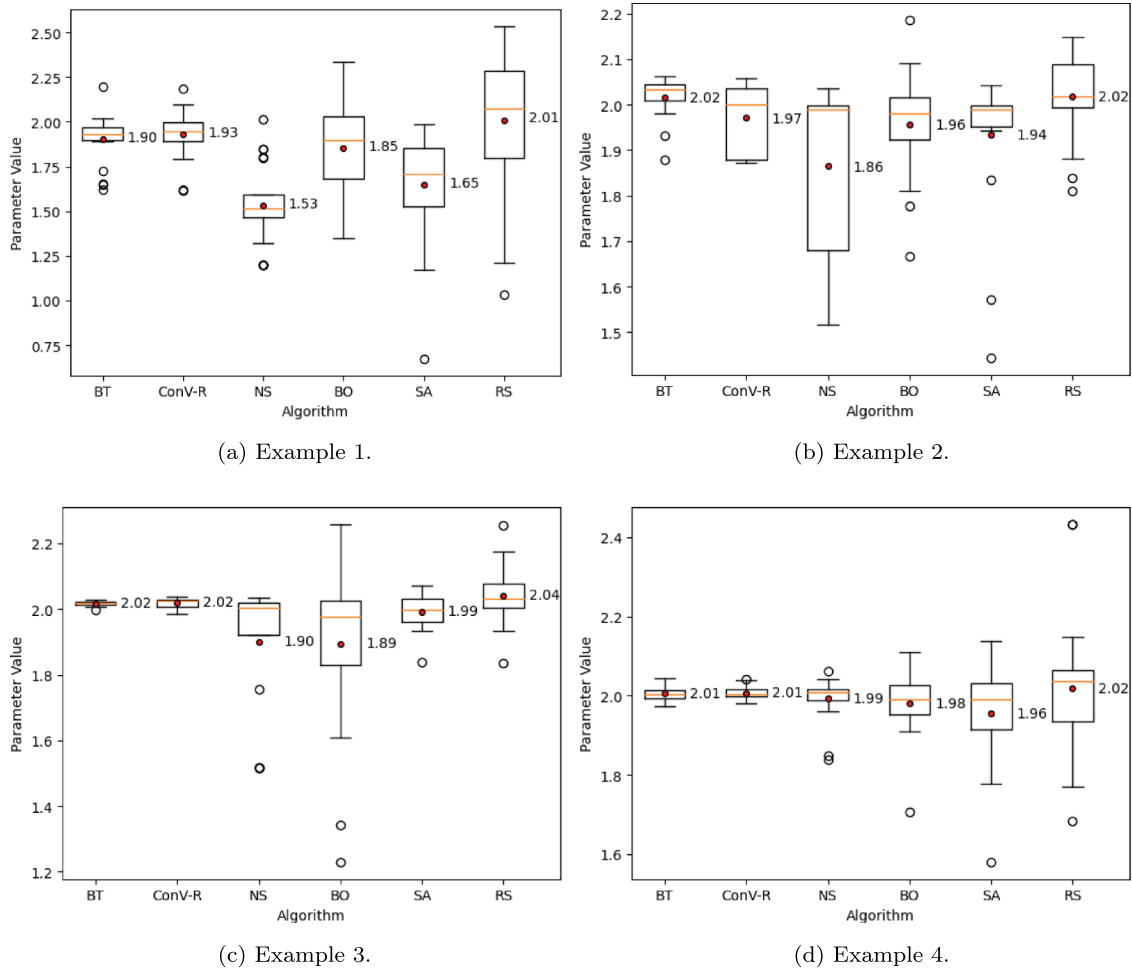
### 5.1. Numerical examples: Static Monte Carlo simulations

To compare the proposed methods with the existing global search methods, we consider the following numerical examples with different characteristics:

- (1) *Example 1: Quadratic model with heterogeneous noise.*
  - *Physical system:*  $Y = (X_1 - 2)^2 + (X_2 - 2)^2 + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, |X_1 X_2 - 2|)$ .
  - *Computer model:*  $h(\theta, X) = (X_1 - \theta)^2 + (X_2 - \theta)^2$ .
- (2) *Example 2: Highly nonlinear model with homogeneous noise.*
  - *Physical system:*  $Y = (X_1 - 2)^7 + (X_2 - 2)^2 + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, 1)$ .
  - *Computer model:*  $h(\theta, X) = (X_1 - \theta)^7 + (X_2 - \theta)^2$ .
- (3) *Example 3: Highly nonlinear model with homogeneous noise and significant difference in scale of one variable against another.*
  - *Physical system:*  $Y = 1000(X_1 - 2)^5 + (X_2 - 2)^2 + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, 1)$ .
  - *Computer model:*  $h(\theta, X) = 1000(X_1 - \theta)^5 + (X_2 - \theta)^2$ .
- (4) *Example 4: Homogeneous noise with interaction terms.*
  - *Physical system:*  $Y = 2X_1 X_2 + \epsilon$ , with  $\epsilon \sim \mathcal{N}(0, 1)$ .
  - *Computer model:*  $h(\theta, X) = \theta X_1 X_2$ .

In all the examples above,  $X = (X_1, X_2)$  where  $X_1 \sim U(0, 4)$ ,  $X_2 \sim U(0, 4)$ , and the optimal parameter value  $\theta^*$  minimizes  $\|h(\theta, X) - Y\|_2^2$ . A dataset of 1,000 data points in each macroreplication is generated and divided into modeling and validation sets with CRN. The optimal parameter for all the examples is 2, and the total budget for all solvers is 1,000 simulation runs. Utilizing the entire dataset to evaluate the objective function for a single  $\theta$  would exhaust the entire budget. Therefore, to enable a comparison with BO, instead of using the entire data for evaluation in each round, we chose a random sample of 50 points for each objective function evaluation. We use the same 50 samples in SA and RS as well. Additionally, in the BO implementation, given that the calibration parameter is one-dimensional, the initial surrogate model is built using a set of 10 randomly selected design points following the recommendation in Loepky et al. (2009), which leads to 10 total BO iterations. SA and RS each ran for 20 iterations. For BT,  $X_1$  and  $X_2$  are considered for stratification. The set of potential concomitant variables considered for ConV are  $\{X_1, X_2, X_1^2, X_2^2, X_1^3, X_2^3\}$ . Since these concomitant variables are chosen from raw data, the approach is denoted as ConV-R.

Figure 1 illustrates how the final solutions vary across the 20 independent runs for each algorithm.



**Figure 1.** Distributions (box plots) of the calibrated parameter values from 20 independent runs of the algorithms in numerical examples. The red dot and the numerical value along each box display the mean value. BT and ConV perform similarly and better than other solvers in mean value and more concentrated distribution. BT exhibits less variability in examples 1 and 2.

Given the limited budget, BT and ConV-R consistently identify the optimal parameter value with minimal variability. In contrast, NS (the same adaptive solver but without stratification) and the global search methods exhibit high variability and sometimes fail to return near-optimal solutions. The high variability implies high risk of these solvers, in the sense that, even if the mean calibrated value is close to the optimal parameter value, a single run of these solvers is more likely to produce a suboptimal solution compared to the proposed approaches. While global methods are expected to converge to the optimal solution with a sufficient computational budget, this experiment shows that the proposed methods can achieve near-optimality faster and more consistently.

Additionally, the local search method without stratification is also prone to high variability or slow convergence compared to the dynamically stratified BT and ConV. BT shows slightly less variability between the two proposed methods in Example 1 and better mean and variance in Example 2. This evidence suggests that BT may be more robust in the presence of heterogeneous noise and more extreme nonlinearity. However, for the latter case, ConV may perform better

with the additional pre-processing to nonlinearly transform  $X_1$  and  $X_2$ .

### 5.2. Calibrating a queuing model: Discrete-event simulation example

In this example, we use calibration to determine the optimal interarrival rate in a simple M/M/1 queue given synthetic data comprising mean service time, mean waiting time, and mean sojourn time. We minimize the discrepancy of the simulated mean waiting time  $h(\theta, X)$  (using interarrival rate  $\theta$ ) and the observed mean waiting time  $Y$ .

The distinction of this example with the static simulations in the previous section is that for ConV method, here we can aggregate a sequence of random variables generated over time and apply Central Limit Theorem (CLT) via standardized mean service time and standardized mean sojourn time as potential concomitant variables (Wilson et al., 1984). The advantage of this transformation is that the ConV method can leverage a lookup table for the optimal strata boundaries of the standardized variables (Jain &

Shashaani, 2023; Sethi, 1963; Wilson et al., 1984) and bypass any additional computation (Step 12–Step 17 in Algorithm 4). As a result, more precision and less clock-time computation in ConV method compared to the BT method may be achieved.

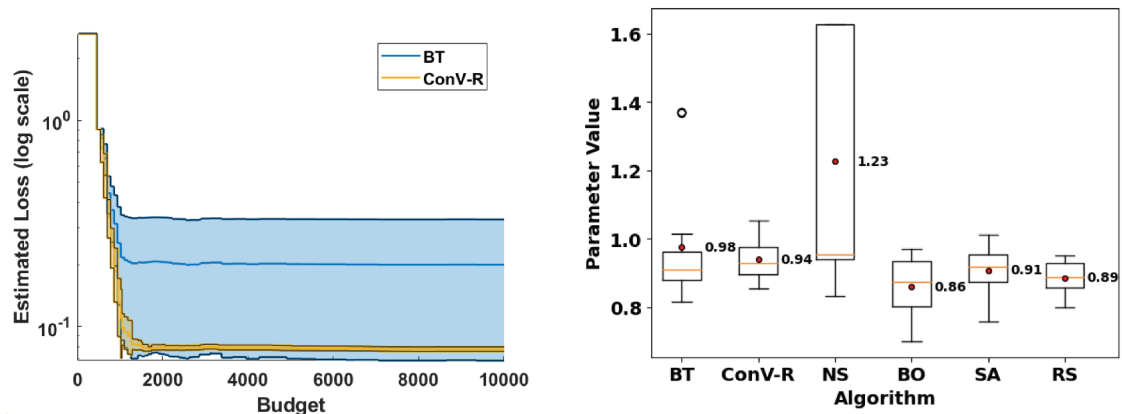
In the experiment, we use a synthetic dataset of 10,000 observations. Each macroreplication starts at the same starting point  $\theta_0 = 1.5$  and has a total budget of 10,000 simulation runs; similar to the previous section, 30% of the dataset in each macro-replication is randomly held for post-processing. The length of the discrete-event queuing simulation is 200 with a warm up period of 50 to obtain a steady state. The service rate is 2, and the optimal interarrival rate (used to generate the waiting and sojourn times in the dataset) is 1. The standardized mean service time and standardized mean sojourn time are utilized as stratification variables for ConV-R. For BT, the mean service time and mean sojourn time are used for stratification. While implementing BO, SA, and RS for comparison, a random sample of 200 points was used in each objective function evaluation.

Figure 2a illustrates the evolution of the objective function across 20 macroreplications as the percentage of expended simulation budget increases. The performance of ConV-R is better than BT as it achieves a lower MSE with minimal variability across macro-replications. This superior performance is anticipated, given that the asymptotic normality of the stratification variables simplifies the implementation of ConV-R and allows for the use of exact strata boundaries derived from theoretical principles without relying on assumptions or error-prone estimations. Importantly, we observe in Figure 2b that the mean of optimal solutions recommended by BT is closer to the optimal parameter; however, the outlier optimal solutions (in the boxplot) and larger variability correspond to

significantly worse objective function values, which is depicted by Figure 2a. The very small variability of the optimal loss values returned by ConV's distribution of optimal solutions suggests that despite their variability, they all yield similar small loss values. Another important observation from Figure 2b is the poor performance of other solvers, above all the NS (same adaptive solver but without stratification) in calibrating the queuing model.

### 5.3. Wind case study: Wake model calibration

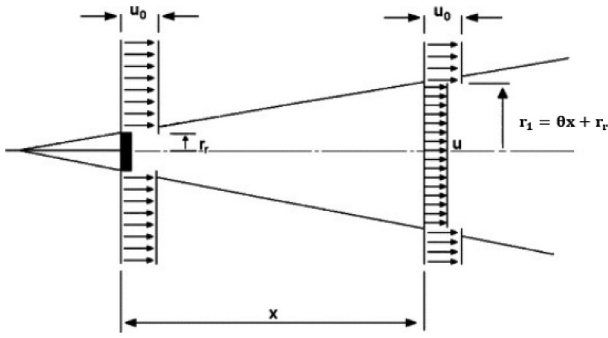
Recall the example we started with in Section 1. The wake effect causes the wind speed reaching the downwind turbines to be less than the wind speed at the upwind turbines, affecting the power generated by these turbines (You, Byon, et al., 2017a, 2017b). Jensen wake model (Jensen, 1983) is a simple but widely used wake model extendable to a multi-turbine setting (Katic et al., 1986) that assumes that wake propagates linearly in the downwind direction, as shown in Figure 3. The value of the wake decay coefficient ( $\theta$  in Figure 3) impacts the performance of the Jensen wake model. Though a value of  $\theta = 0.04$  is widely assumed for offshore wind farms (Barthelmie et al., 2010; Katic et al., 1986), some recent studies have shown that this value does not necessarily depict the wind speed reduction observed in actual wind farms (Göçmen et al., 2016; You, Liu, et al., 2017b). Thus, it is essential to determine the value of this wake decay coefficient for each wind farm separately. The wake model simulates the wind speed at each turbine in the wind farm. The power curve for the turbines is generated via B-splines using the data at one of the upwind turbines (Lee et al., 2013; You, Liu, et al., 2017). This power curve is used to estimate the power generated at each turbine.



(a) 95% CI of progress curves generated from the proposed approaches for the queuing calibration reveals that ConV-R performs significantly better than BT with lower optimal loss and much less variability.

(b) Distribution of the calibrated parameter values from 20 independent runs of the algorithm for the queuing calibration. The red dot and the numerical value along the boxplot display the mean value. BT and ConV-R are more accurate with smaller variances.

Figure 2. Comparison of the performance of solvers for the queuing model calibration.



**Figure 3.** Linear propagation of wake as modeled by the Jensen wake model, where  $r_r$  is the rotor radius and  $u_0$  is the free-stream wind speed (excerpted from Jensen (1983)).

In our case study, data is collected from an offshore wind farm with 30+ turbines. The data includes information about wind conditions, such as the 10-min average wind speed (WS) and direction, turbulence intensity (TI), etc. Along with this, it also consists of a 10-min average power generated by each turbine. In the analysis, the power generated by each turbine is normalized by dividing it by the maximum possible power that can be generated, referred to as nominal power (Byon et al., 2011). For a given combination of input wind condition  $X$  (WS, TI, etc.) and the wake decay coefficient  $\theta$ , Jensen's wake model estimates the power generated by turbines  $h(\theta; X)$ . This simulated power is then compared to the observed power at turbines  $Y$  to get  $F(\theta, (X, Y))$ , the objective function value, the loss function measuring the discrepancy between observed power and model predicted ones.

### 5.3.1. Implementation

A modeling set comprising 70% data used for optimization is sampled independently for each macro-replication. Each macro-replication starts at the same initial point  $\theta_0 = 0.1$ , the initial TR radius  $\Delta_0 = 0.08$  and the minimum sample size  $\lambda_0 = 80$ , and runs for a total of 10,000 simulations (budget). In our first proposed approach, the input variables WS and TI are used as the stratification variables for dividing the data via binary trees (BT).

In our second proposed approach, two cases are considered for stratification with concomitant variables: using real data (ConV-R) or the simulated data (ConV-S). In the first method, we consider five alternatives to stratify the data using input WS and TI along with nonlinear transformations  $WS^2$ ,  $TI^2$ , and  $WS^3$ . With unknown joint probability distribution of TI and WS, the strata boundaries are determined by solving the iterative method using the population  $\mathcal{X}$ . Based on the stratification boundaries, the real data is divided into non-overlapping sets  $\mathcal{X}_{k,z}$ ,  $z = 1, 2, \dots, Z_k$ , and the probabilities are  $p_{k,z} = |\mathcal{X}_{k,z}|/|\mathcal{X}|$ . For a given  $\theta_k$ , the Jensen model simulates the wind speeds reaching each

turbine, providing the mean estimated wind speed at the turbines  $\widehat{WS}_k$ . The model also provides the simulated power at each turbine using this simulated wind speed and the power curve. Thus, another possibility of a concomitant variable is the mean estimated power at the turbines  $\hat{h}(\theta_k, X)$ . For stratification using simulated data (ConV-S), we consider these two variables along with their nonlinear transformations  $\widehat{WS}_k^2$ ,  $\hat{h}^2(\theta_k, X)$ , and  $\widehat{WS}_k^3$ . When using these simulated variables for stratification, the strata boundaries are determined by using the iterative method with  $\lambda_k$  points, and the probabilities are estimated as  $p_{k,z} \approx \lambda_{k,z}/\lambda_k$ . For both ConV-R and ConV-S at each iteration, the variable with the lowest residual variance is chosen as the concomitant variable. Thus, we do not choose a concomitant variable a priori; the algorithm identifies it adaptively.

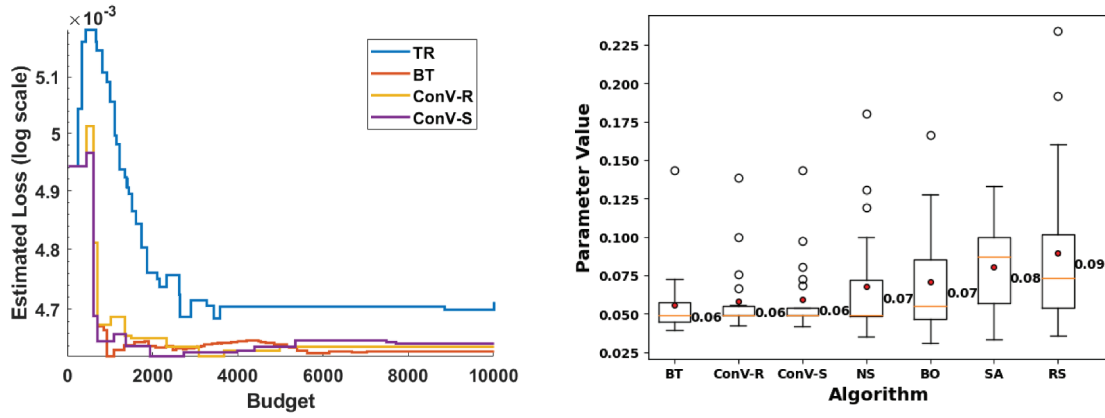
### 5.3.2. Results

Figure 4a compares how the expected progress varies during optimization for the no stratification case (NS) and the solvers with dynamic stratification all under ASTRO-DF optimization algorithm. The main advantage of using the proposed stratified adaptive methods is a significant improvement in performance initially to reach better solutions. All of the three proposed approaches provide comparable results. BT and ConV-R exhibit more similar performance, which is interesting given that ConV-R uses only one variable at a time for stratification. Another observation is that they both reach better solutions compared to ConV-S, which is expected as ConV-S builds the stratification structure with a small sample of noisy simulated data. The second advantage is depicted by Figure 5a-c where the 95% CI widths of BT, ConV-R, and ConV-S are smaller, indicating reduced variability or uncertainty (risk) in the performance of the optimization algorithm.

Figure 4b compares the performance of BT and ConV against the global search algorithms (BO, SA, and RS) and the non-stratified version of the adaptive local solver (NS). The stratification methods show much lower variability. Although the mean calibrated parameter values for BT and ConV are similar, BT exhibits more variability (larger interquartile range

Recall that ConV-R and ConV-S dynamically identify the best concomitant variable throughout iterations. Table 1 summarizes the mean frequency with which a concomitant variable is chosen for the baseline case  $\theta_0 = 0.1$ ,  $\Delta_0 = 0.08$ , and  $\lambda_0 = 80$ . For ConV-R, TI and its squared transformation are often picked for stratification. In the literature, the wake decay coefficient has been shown to correlate well with TI (Barthelmie et al., 2015; Duc et al., 2019; Peña et al., 2016). Thus, consistently choosing TI by the algorithm indicates that it is aptly choosing the best stratification variable. In ConV-

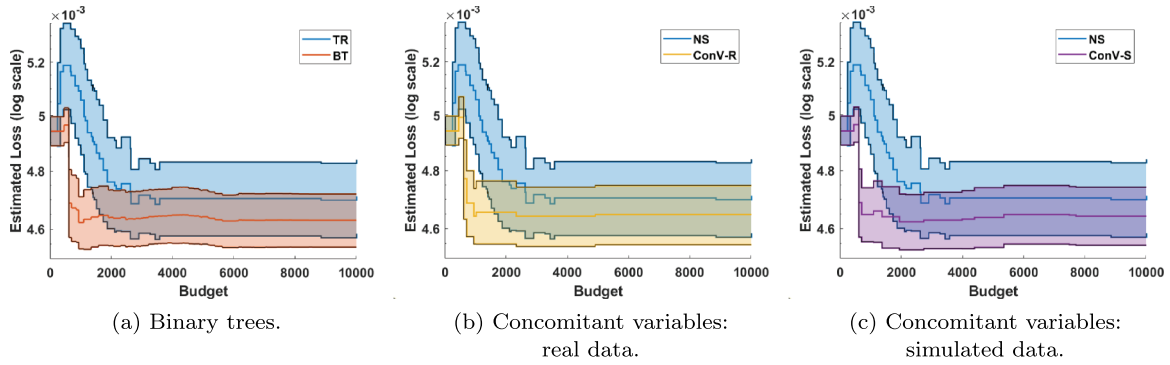




(a) Progress of the mean objective function value during optimization over 20 macro-replications for  $\theta_0 = 0.1$ ,  $\Delta_0 = 0.08$ , and  $\lambda_0 = 80$ . The x-axis represents the number of simulations.

(b) Distribution of the calibrated parameter values from 20 independent runs of the algorithm for the wind case study. The red dot and the numerical values along the boxplot displays the mean value.

**Figure 4.** Comparison of the performance of solvers for the wake model calibration. BT and ConV perform better than the global solvers.



**Figure 5.** Variability and risk (95% CI progress curves) of proposed approaches (BT, ConV-r, and ConV-s) is reduced compared to no-stratification (NS), computed over 20 macro-replications.

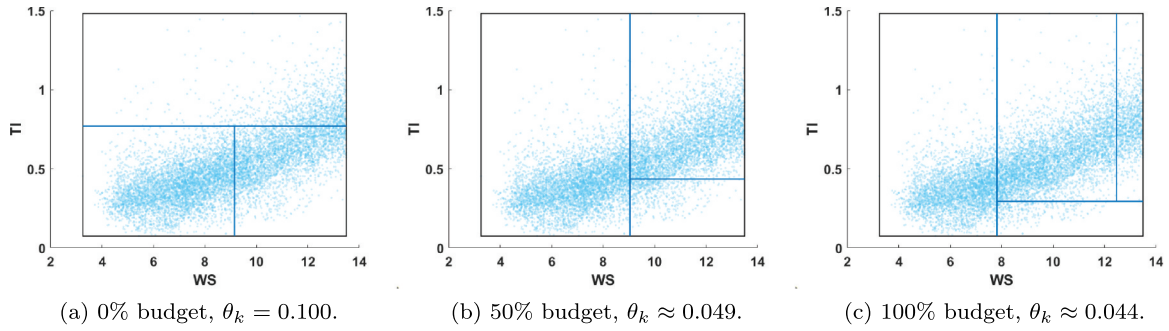
**Table 1.** Mean frequency with which a particular variable is picked for stratification across 20 macro-replications. Note that these values are for  $\theta_0 = 0.1$ ,  $\Delta_0 = 0.08$ , and  $\lambda_0 = 80$ . The distribution can change with the changes in these initial settings (the value in the parenthesis is the standard error).

ConV-R	Variables	WS	TI	$WS^2$	$TI^2$	$WS^3$
	Frequency	0.05(0.05)	11.20(2.36)	0.35(0.30)	24.50(3.48)	3.00(1.55)
ConV-S	Variables	$\widehat{WS}_k^2$	$\hat{h}(\theta_k, X)$	$\widehat{WS}_k^2$	$\hat{h}^2(\theta_k, X)$	$\widehat{WS}_k^3$
	Frequency	2.00(1.05)	0.30(0.22)	0.20(0.14)	32.50(2.51)	1.10(0.45)

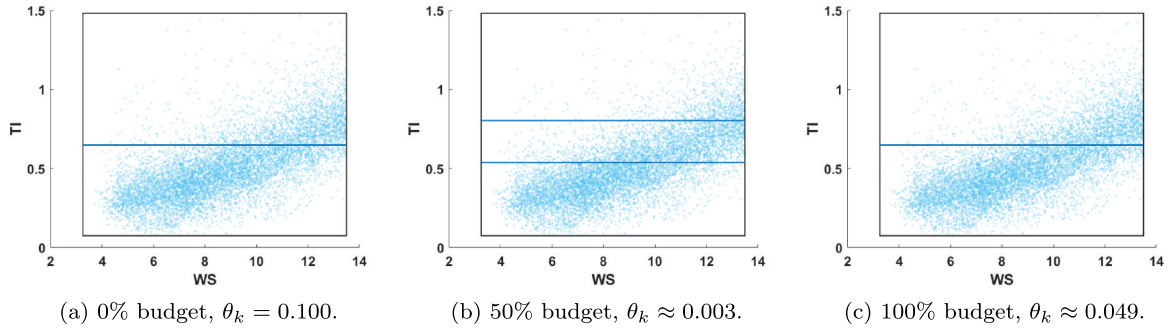
S, the mean of the squared simulated power at each turbine is chosen almost every time. The loss function is mathematically more correlated with  $\hat{h}^2(\theta_k, X)$  than any other variable. Hence, choosing it consistently again indicates that the proposed method can determine the best stratification variable.

Figures 6 and 7 show how the stratification structure changes during optimization when using BT and ConV-R, respectively. Unlike stratification with concomitant variables, binary trees can divide the data based on multiple variables (TI and WS), as shown in Figure 6a-c. While computationally more intensive, this method is more flexible in choosing

the stratification variable and deciding the number of strata. Recall that in BT, real data corresponding to  $\lambda_k$  pilot simulations is used for stratification, and in ConV-R, the entire data is used for stratification. If the number of strata and the stratification variable are the same, ConV-R will generate the same structure irrespective of  $\theta_k$  as seen in Figure 7 where the strata design for  $\theta_k = 0.100$  and  $\theta_k = 0.049$  is the same. However, the number of strata and the stratification variable depends on  $\theta_k$ , which makes the stratification dynamic in ConV-R. Additionally, the choices for the stratification structure throughout the optimization are finite (for each possible



**Figure 6.** Evolution of the stratification structure, within BT, during optimization for a single macro-replication after approximately 0%, 50%, and 100% budget is utilized. The points denote the actual data.



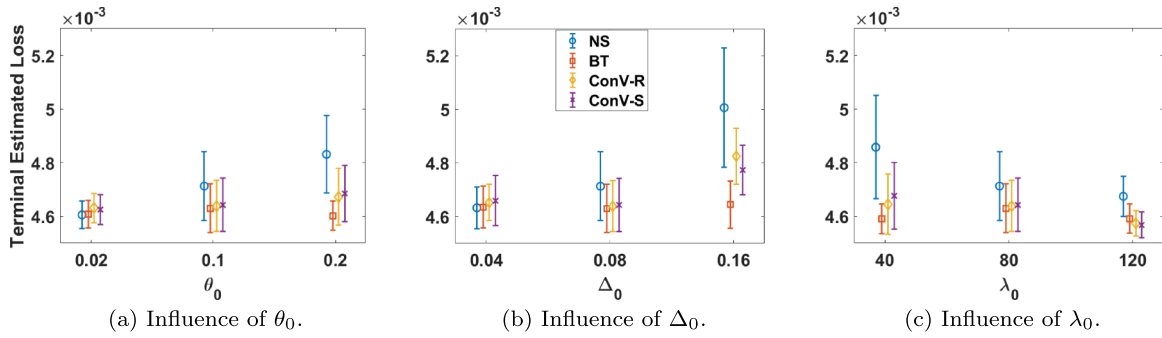
**Figure 7.** Evolution of the stratification structure, within ConV-r, during optimization for a single macro-replication after approximately 0%, 50%, and 100% budget is utilized. The points denote the actual data.

concomitant variable and each possible number of strata), which can reduce the run-to-run variability of the algorithm compared to other cases where there may be virtually infinite choices for the stratification structure.

Next, we compare the robustness of the proposed methods with a Box-Wilson Central Composite Design (CCD). A CCD provides enough information to estimate the main effects and interactions with significantly fewer designs than a full-factorial design (Hill & Hunter, 1966). We test the proposed methods' sensitivity by varying the algorithm's three most critical hyperparameters,  $\theta_0$ ,  $\Delta_0$ , and  $\lambda_0$ . Considering  $\theta_0 = 0.1$ ,  $\Delta_0 = 0.08$ , and  $\lambda_0 = 80$  as the baseline case, the robustness is analysed by fixing two parameters and perturbing the third between two relatively extreme values. We consider the following parameter values for the analysis:  $\theta_0 = \{0.02, 0.2\}$ ,  $\Delta_0 = \{0.04, 0.16\}$ , and  $\lambda_0 = \{40, 80\}$ . *Detailed hyperparameter tuning is beyond the scope of this paper; the ranges selected are reasonable for each parameter in the context of this problem and the objective of this sensitivity analysis study is to examine whether the performance of the proposed algorithms would significantly vary for different starting conditions.* Figure 8 depicts the error bars of each solver's terminal objective function values obtained from 20 macro-replications, considering various designs. In summary, efficient dynamic stratification diminishes the reliance of TR algorithms on hyperparameters, enhancing their robustness.

Figure 8a investigates the influence of initial solution  $\theta_0$  on the performance of the solvers. With a favourable starting point,  $\theta_0 = 0.02$  (where we speculate that the objective function is at a steep region), the performance across all cases is about the same. This observation aligns with expectations, as the proximity of the starting point to the true optimum allows the algorithms to reach the optimal solution with minimal exploration. Conversely, for  $\theta_0 = 0.2$ , where the starting point is considerably far from the true optimum and at a more flat region, NS exhibits significantly worse performance than using dynamic strata, highlighting that changing strata effectively can lead to robust exploration and, thus, better performance.

Figure 8b illustrates the effect of the initial TR radius,  $\Delta_0$ . A larger  $\Delta_0$  facilitates early exploration and demands that the solvers execute efficient exploitation. Failure to accomplish this may lead to the algorithm becoming trapped in a suboptimal region. This is particularly evident in the case of NS, where its performance deteriorates with increasing  $\Delta_0$ . In contrast, dynamic strata enhance early exploitation to a certain degree, enabling the solvers to attain improved solutions. Stratification with concomitant variables shows some sensitivity to the initial TR radius, degrading their performance for larger  $\Delta_0$  values. The enhanced flexibility provided by BT, allowing the selection of multiple stratification



**Figure 8.** Effect of different hyperparameters on the performance of the solvers. Implementing stratification reduces the algorithm’s dependence on the choice of the hyperparameters (baseline setting:  $\theta_0 = 0.1$ ,  $\Delta_0 = 0.08$ , and  $\lambda_0 = 80$ ).

variables simultaneously, may contribute to improved early exploitation, potentially explaining its performance for  $\Delta_0 = 0.16$ .

Figure 8c demonstrates how the initial sample size,  $\lambda_0$ , influences the solver’s performance. A small  $\lambda_0$  allows the algorithm more budget for exploration but can lead to imprecise estimates and, consequently, poor exploitation. As  $\lambda_0$  increases, the performance of all solvers generally improves, but for a limited budget setting, a large  $\lambda_0$  may not be preferable. Stratified sampling becomes crucial in this context as it provides more accurate estimates for smaller sample sizes. This capability allows solvers employing dynamic stratified sampling to outperform others, even when  $\lambda_0$  is small.

#### 5.4. Discussion

While adaptive sampling in NS efficiently allocates the simulation budget for each  $\theta$  based on its unconditional output variance and proximity to optimality, stratified sampling further enhances efficiency by reducing output variance with conditioning and allowing the sample size stopping conditions to be met earlier. The effectiveness of stratified sampling depends on the stratification structure, which optimally partitions the input space to minimize output variance. Since output variance structure can significantly vary from one system (calibration parameter) to another – heteroscedasticity (with respect to  $\theta$ ), the proposed dynamically stratified adaptive sampling procedure aims to learn about the local variance structure of the output to maximize efficiency. Any optimal budget allocation during optimization can enhance exploitation by improving estimation error with fewer samples (simulation runs) at each evaluation. Thrifty exploitation ensures ample budget is saved for exploration and allows the algorithm to run for more iterations. Therefore, a dynamically stratified adaptive sampling procedure increases the solver’s ability to more thoroughly explore the decision space.

In our experiments, the benefit of stratification is evident across all examples, consistently outperforming the non-stratified approach and the global solvers, as the recommended parameter values using dynamically stratified adaptive sampling are almost always closer to the true optimal and more consistent (less risky) across independent solver runs. Such advantages are notable, especially because the stratification procedure is not very time-consuming as the additional time needed for stratification is negligible compared to each simulation run. For example, in our queuing experiments, the average clock time to solve with NS (no stratification) was 142.0 s, while that of the BT and ConV solvers was 146.1 and 147.8 s, respectively. The rewards in finite-time solution performance easily justifies the added  $\sim 4\%$  increase in clock time. The computational overhead for stratification becomes more negligible when the simulation runtime is longer, such as wind power simulation.

Which dynamic stratification method should one choose? The answer to this question is contingent upon the structural characteristics of the problem. We highlight the following observations that can aid in incorporating these dynamically stratified adaptive sampling procedures within a solver:

- (i) While in most cases the two approaches perform similarly, Example 2 in Section 5.1 suggests that BT may perform better than ConV in extreme nonlinearity or interactions (dependencies) in input variables.
- (ii) For ConV to work well in these situations, additional pre-processing to find a good non-linear transformation of the input variables may be necessary. This is because ConV relies on a linear mapping between what it will use as the concomitant variable and the objective (loss) function value  $F$ . At the minimum, the squared transformations of the input variables should be considered in calibration problems

with MSE-like loss functions. How to find more linearly related concomitant variables inexpensively and whether that effort may be worthwhile is an open research question.

- (iii) ConV has another restriction in only choosing one variable to stratify each time. At the time of writing this paper, we are unsure of whether that restriction necessarily translates to a weakness since in more extensive experimentation that we did not include in this paper, ConV performed competitively with BT for higher-dimensional and more inter-dependent input spaces or cases where multiple stratification variables held significance. An explanation for this observation may be that by the parsimony principles (Goloboff, 2003), finding the single input variable that is the major contributor to heterogeneity of output variance in the input space for a fixed  $\theta$  may be sufficient and less prone to statistical errors and biases when forming the strata. This point is visible in the wind power calibration case study in Figure 4b, where the BT boxplot of optimal solutions is wider compared to ConV and we see in Figure 6 that BT sometimes stratifies with more than one input variable.
- (iv) Our sensitivity analysis with real data suggests that BT can have a slightly more robust performance compared to ConV with respect to the solver's starting conditions (initial solution, minimum sample size, initial step size); yet a more extensive experimental design to make a general judgement on sensitivity is left for future research.
- (v) ConV becomes efficient when the distribution of the stratification variable is either known or can be approximated, as seen in Section 5.2 where the distribution of standardized mean service time and the standardized mean sojourn time can be approximated. For most of the time-dependent simulations, it is easy to use CLT to approximate the distribution of variables, and for these cases, stratification with concomitant variables can be very effective.
- (vi) In using ConV, if the concomitant variables are chosen from the real (not simulated) input data, establishing strata requires minimal computation. More importantly, using all of the real data does not affect the simulation budget and significantly reduces the inherent noise when statistics from each strata is used to estimate the objective function or the sample size.

## 6. Conclusion

In data-driven calibration, the presence of abundant data with many covariates can help match the model

outputs and observed outputs by tuning the model parameters. To reduce computation, using subsamples of data makes the problem stochastic and apt for simulation optimization, in which a vast amount of joint information can aid using stratified sampling to reduce estimation error at each visited calibration parameter. However, stratified sampling within simulation optimization is challenging. We propose using post-stratification to lower the instability of sampling distributions throughout the optimization. This stability enables a more tailored design of strata that, if done at a low cost, has the potential to save exploitation sampling efforts for more exploration in the search.

We further propose two ways for dynamic stratification. The first approach determines strata boundaries by hierarchically dividing the data using binary trees that are grown only until enough information can be gained. This approach may be computationally expensive but is flexible as it can concurrently use multiple-stratification variables. In the second approach, concomitant variables help form strata. If these variables exhibit a positive correlation with the simulation output, they can be employed to establish optimal strata boundaries through closed-form equations. Using pilot simulation runs, we propose methods to find the best concomitant variables (that can be nonlinear transformations of real inputs or generated data during each simulation run) and the number of strata for this purpose. However, the strata can be formed with less dependence on limited runs and less computation by leveraging population statistics or looking up exact values by studentizing variables that store aggregated information. A comparison case study on the real-world data for a wind power model calibration and some static and time-dependent simulation examples illustrates faster progress and less run-to-run solver variability. Effective stratification further reduces the solver's reliance on hyperparameters, making it more robust.

While both approaches show similar performance in many cases, choosing one approach over the other hinges on the nature of the relationship between stratification variables and the simulation output, the significance of multiple stratification variables, the availability of information regarding the distribution of these variables, and the presence of noise in simulations. As a rule of thumb, if there is evidence for heterogeneous noise and non-linear relationships that may take time to unravel, then binary trees may be more beneficial. On the other hand, if, from expert opinion or descriptive analysis, we can find or form a concomitant variable that is linearly dependent on the response, it may single-handedly help partition the input space to tackle heteroscedasticity at a low computational cost. Among choices for concomitant variables, choosing the real data instead of simulated data may be more effective, particularly when simulation outputs are too noisy.

The present study centres on minimizing variance, which is the goal of stratified sampling in estimation



and inference. However, in optimization, variance may only need to be reduced so much to help with the progress. In other words, the cost of maximal variance reduction may waste too much of the computational budget. Therefore, future work will view stratified sampling for optimization with a different objective in mind: making better local approximations that guarantee just enough accuracy to economize budget expenditure in the early iterations. Particularly for a class of adaptive simulation optimization solvers, this road-map can lead to proven lower sample complexity that can be fundamental to the theory and application of simulation optimization solvers for digital twins (Goodwin et al., 2022; Santos et al., 2022). Deriving closed-form equations for simultaneously utilizing multiple concomitant variables for stratification is also an unexplored area for future research. Furthermore, stratification with concomitant variables requires pre-processing to identify the appropriate function of the concomitant variable that has a linear relationship with the output. This aspect is left for future research.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

The work was supported by the National Science Foundation [CMMI-2226347 and CMMI-2226348] as well as Office of Naval Research [N000142412398].

## ORCID

Sara Shashaani  <http://orcid.org/0000-0001-8515-5877>

## References

- Aguiar, N. R., Neto, Á. B., Bezerra, Y. S. D. F., Do Nascimento, H. A. D., Lucena, L. D. S., de Freitas, J. E., & Mirjalili, S. (2022). A new hybrid optimization approach using pso, nelder-mead simplex and kmeans clustering algorithms for 1d full waveform inversion. *PLOS ONE*, 17(12), e0277900. <https://doi.org/10.1371/journal.pone.0277900>
- Andradóttir, S. (2006). Chapter 20 An Overview of Simulation Optimization via Random Search. In G. H. Shane & L. N. Barry (Eds.), *Simulation*. Handbooks in Operations Research and Management Science (Vol. 13, pp. 617–631). Elsevier. [https://doi.org/10.1016/S0927-0507\(06\)13020-0](https://doi.org/10.1016/S0927-0507(06)13020-0)
- Andradóttir, S., & Prudius, A. A. (2009). Balanced explorative and exploitative search with estimation for simulation optimization. *INFORMS Journal on Computing*, 21(2), 193–208. <https://doi.org/10.1287/ijoc.1080.0309>
- Asmussen, S., & Glynn, P. W. (2007). *Stochastic simulation: Algorithms and analysis* (Vol. 57). Springer.
- Barthelmie, R. J., Churchfield, M. J., Moriarty, P. J., Lundquist, J. K., Oxley, G., Hahn, S., & Pryor, S. (2015). The role of atmospheric stability/turbulence on wakes at the Egmond aan zee offshore wind farm. *Journal of Physics: Conference Series*, 625, 012002. <https://doi.org/10.1088/1742-6596/625/1/012002>
- Barthelmie, R. J., Pryor, S. C., Frandsen, S. T., Hansen, K. S., Schepers, J., Rados, K. . . . Neckelmann, S. (2010). Quantifying the impact of wind turbine wakes on power output at offshore wind farms. *Journal of Atmospheric and Oceanic Technology*, 27(8), 1302–1317. <https://doi.org/10.1175/2010JTECHA1398.1>
- Bretthauer, K. M., Ross, A., & Shetty, B. (1999). Nonlinear integer programming for optimal allocation in stratified sampling. *European Journal of Operational Research*, 116(3), 667–680. [https://doi.org/10.1016/S0377-2217\(98\)00180-5](https://doi.org/10.1016/S0377-2217(98)00180-5)
- Brito, J., Maculan, N., Lila, M., & Montenegro, F. (2010). An exact algorithm for the stratification problem with proportional allocation. *Optimization letters*, 4(2), 185–195. <https://doi.org/10.1007/s11590-009-0157-2>
- Burden, R. L., & Faires, J. D. (1989). 2.2 fixed-point iteration. In Prindle, Weber, & Schmidt (Eds.), *Numerical analysis*. Prindle, Weber & Schmidt series in mathematics (4th ed., Vol. 1, p. 729). PWS-Kent Publishing Company.
- Byon, E., Pérez, E., Ding, Y., & Ntamo, L. (2011). Simulation of wind farm operations and maintenance using discrete event system specification. *Simulation*, 87(12), 1093–1117. <https://doi.org/10.1177/0037549710376841>
- Byrd, R. H., Chin, G. M., Nocedal, J., & Wu, Y. (2012). Sample size selection in optimization methods for machine learning. *Mathematical Programming*, 134(1), 127–155. <https://doi.org/10.1007/s10107-012-0572-5>
- Chen, A. A., Chai, X., Chen, B., Bian, R., & Chen, Q. (2018). A novel stochastic stratified average gradient method: Convergence rate and its complexity. In *2018 International Joint Conference on Neural Networks (IJCNN)*, Rio de Janeiro, Brazil (pp. 1–8).
- Cheng, R., Dye, C., Dagpunar, J., & Williams, B. (2023). Modelling presymptomatic infectiousness in covid-19. *Journal of Simulation*, 17(5), 532–543. <https://doi.org/10.1080/17477778.2023.2190467>
- Cochran, W. G. (1977). *Sampling techniques*. John Wiley & Sons.
- Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to derivative-free optimization*. SIAM.
- Dalenius, T. (1950). The problem of optimum stratification. *Scandinavian actuarial journal*, 1950(3–4), 203–213. <https://doi.org/10.1080/03461238.1950.10432042>
- Dalenius, T., & Gurney, M. (1951). The problem of optimum stratification. ii. *Scandinavian actuarial journal*, 1951(1–2), 133–148. <https://doi.org/10.1080/03461238.1951.10432134>
- de Moura Brito, J. A., Semaan, G. S., Fadel, A. C., & Brito, L. R. (2017). An optimization approach applied to the optimal stratification problem. *Communications in Statistics - Simulation and Computation*, 46(6), 4419–4451. <https://doi.org/10.1080/03610918.2015.1118505>
- Duc, T., Coupiac, O., Girard, N., Giebel, G., & Göçmen, T. (2019). Local turbulence parameterization improves the Jensen wake model and its implementation for power optimization of an operating wind farm. *Wind Energy Science*, 4(2), 287–302. <https://doi.org/10.5194/wes-4-287-2019>
- Eckman, D. J., Henderson, S. G., & Shashaani, S. (2023). Diagnostic tools for evaluating and comparing simulation-optimization algorithms. *INFORMS Journal on Computing*, 35(2), 350–367. <https://doi.org/10.1287/ijoc.2022.1261>

- Espath, L., Krumscheid, S., Tempone, R., & Vilanova, P. (2021). On the equivalence of different adaptive batch size selection strategies for stochastic gradient descent methods. *arXiv Preprint arXiv*. <https://doi.org/10.48550/arXiv.2109.10933> accessed 23rd May 2022.
- Etoré, P., Fort, G., Jourdain, B., & Moulines, E. (2011). On adaptive stratification. *Annals of Operations Research*, 189(1), 127–154. <https://doi.org/10.1007/s10479-009-0638-9>
- Etoré, P., & Jourdain, B. (2010). Adaptive optimal allocation in stratified sampling methods. *Methodology and Computing in Applied Probability*, 12(3), 335–360. <https://doi.org/10.1007/s11009-008-9108-0>
- Farias, F., Ludermir, T., & Bastos-Filho, C. (2020). Similarity based stratified splitting: An approach to train better classifiers. *arXiv Preprint arXiv: 2010.06099*. <https://doi.org/10.48550/arXiv.2010.06099> accessed 3rd May 2022.
- Frazier, P. I. (2018). Bayesian optimization. In G. Esma & N. Lewis (Eds.), *Recent advances in optimization and modeling of contemporary problems* (pp. 255–278). Informa.
- Gao, S., Lee, L. H., Chen, C.-H., & Shi, L. (2015). A sequential budget allocation framework for simulation optimization. *IEEE Transactions on Automation Science and Engineering*, 14(2), 1185–1194.
- Glynn, P. W., & Zheng, Z. (2021). Efficient computation for stratified splitting. In S. Kim. (Ed. *Proceedings of the 2021 winter simulation conference* (pp. 1–8). Piscataway, New Jersey.
- Göçmen, T., Van der Laan, P., Réthoré, P.-E., Diaz, A. P., Larsen, G. C., & Ott, S. (2016). Wind turbine wake models developed at the technical university of Denmark: A review. *Renewable and Sustainable Energy Reviews*, 60, 752–769. <https://doi.org/10.1016/j.rser.2016.01.113>
- Goloboff, P. A. (2003). Parsimony, likelihood, and simplicity. *Cladistics*, 19(2), 91–103. <https://doi.org/10.1111/j.1096-0031.2003.tb00297.x>
- Goodwin, T., Xu, J., Celik, N., & Chen, C.-H. (2022). Real-time digital twin-based optimization with predictive simulation learning. *Journal of Simulation*, 18(1), 47–64. <https://doi.org/10.1080/17477778.2022.2046520>
- Guzmán-Cruz, R., Castañeda-Miranda, R., García-Escalante, J. J., López-Cruz, I., Lara-Herrera, A., & De la Rosa, J. (2009). Calibration of a greenhouse climate model using evolutionary algorithms. *Biosystems Engineering*, 104(1), 135–142. <https://doi.org/10.1016/j.biosystemseng.2009.06.006>
- Ha, Y., & Shashaani, S. (2023). Iteration complexity and finite-time efficiency of adaptive sampling trust-region methods for stochastic derivative-free optimization. *arXiv Preprint arXiv: 2305.10650*, 1–15. <https://doi.org/10.1080/24725854.2024.2335513>
- Ha, Y., Shashaani, S., & Pasupathy, R. (2024). Complexity of Zeroth-and First-order Stochastic Trust-Region Algorithms. *arXiv preprint arXiv:2405.20116*.
- Hassan, A., Abdel-Malek, H., & Rabie, A. (2006). Non-derivative design centering algorithm using trust region optimization and variance reduction. *Engineering Optimization*, 38(1), 37–51. <https://doi.org/10.1080/03052150500323880>
- Hill, W. J., & Hunter, W. G. (1966). A review of response surface methodology: A literature survey. *Technometrics*, 8(4), 571–590. <https://doi.org/10.2307/1266632>
- Holland, P. W., & Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics - Theory and Methods*, 6(9), 813–827. <https://doi.org/10.1080/03610927708827533>
- Huddleston, H., Claypool, P., & Hocking, R. (1970). Optimal sample allocation to strata using convex programming. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 19(3), 273–278. <https://doi.org/10.2307/2346332>
- Jain, P., & Shashaani, S. (2023). Dynamic stratification and post-stratified adaptive sampling for simulation optimization. In *2023 Simulation Conference (WSC)*, San Antonio, TX, USA (pp. 3460–3471).
- Jain, P., Shashaani, S., & Byon, E. (2021). Wake effect calibration in wind power systems with adaptive sampling based optimization. In A. Ghate, K. Krishnaiyer, & K. Paynabar (Eds.), *Proceedings of the IISE annual conference* (pp. 43–48). Red Hook, NY.
- Jain, P., Shashaani, S., & Byon, E. (2022). Robust simulation optimization with stratification. In B. Feng. (Ed.), *Proceedings of the 2022 Winter Simulation Conference*, Piscataway, New Jersey.
- Jain, P., Shashaani, S., & Byon, E. (2023). Wake effect parameter calibration with large-scale field operational data using stochastic optimization. *Applied Energy*, 347, 121426. <https://doi.org/10.1016/j.apenergy.2023.121426>
- Jensen, N. O. (1983). *A note on wind generator interaction*. National Laboratory.
- Jeong, C., & Byon, E. (2024). Calibration of building energy computer models via bias-corrected iteratively reweighted least squares method. *Applied Energy*, 360, 122753. <https://doi.org/10.1016/j.apenergy.2024.122753>
- Jeong, C., Xu, Z., Berahas, A. S., Byon, E., & Cetin, K. (2023). Multiblock parameter calibration in computer models. *INFORMS Journal on Data Science*, 2(2), 116–137. <https://doi.org/10.1287/ijds.2023.0029>
- Juan Felipe Parra, P. J., & Arango-Aramburo, S. (2018). Metaheuristic optimization methods for calibration of system dynamics models. *Journal of Simulation*, 12(2), 190–209. <https://doi.org/10.1080/17477778.2018.1467850>
- Katic, I., Højstrup, J., & Jensen, N. O. (1986). A simple model for cluster efficiency. In *European wind energy association conference and exhibition*, Rome, Italy (Vol. 1, pp. 407–410).
- Kawai, R. (2010). Asymptotically optimal allocation of stratified sampling with adaptive variance reduction by strata. *ACM Transactions on Modeling and Computer Simulation*, 20(2), 1–17. <https://doi.org/10.1145/1734222.1734225>
- Kennedy, M. C., & O’Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society Series B, Statistical Methodology*, 63(3), 425–464. <https://doi.org/10.1111/1467-9868.00294>
- Keskintürk, T., & Er, S. (2007). A genetic algorithm approach to determine stratum boundaries and sample sizes of each stratum in stratified sampling. *Computational Statistics & Data Analysis*, 52(1), 53–67. <https://doi.org/10.1016/j.csda.2007.03.026>
- Khan, M. G. M., Nand, N., & Ahmad, N. (2008). Determining the optimum strata boundary points using dynamic programming. *Survey Methodology*, 34(2), 205–214.
- Lee, G., Byon, E., Ntamo, L., & Ding, Y. (2013). Bayesian spline method for assessing extreme loads on wind turbines. *The Annals of Applied Statistics*, 7(4), 2034–2061. <https://doi.org/10.1214/13-AOAS670>
- Liu, B., Yue, X., Byon, E., & Kontar, R. A. (2022). Parameter calibration in wake effect simulation model with stochastic gradient descent and stratified sampling. *The Annals of Applied Statistics*, 16(3), 1795–1821. <https://doi.org/10.1214/21-AOAS1567>
- Liu, S., Butler, D., Brazier, R., Heathwaite, L., & Khu, S.-T. (2007). Using genetic algorithms to calibrate a water

- quality model. *Science of the Total Environment*, 374 (2–3), 260–272. <https://doi.org/10.1016/j.scitotenv.2006.12.042>
- Loeppky, J. L., Sacks, J., & Welch, W. J. (2009). Choosing the sample size of a computer experiment: A practical guide. *Technometrics*, 51(4), 366–376. <https://doi.org/10.1198/TECH.2009.08040>
- Neyman, J. (1934). On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4), 558–625. Retrieved 2022-07-19, from <http://www.jstor.org/stable/2342192>
- Peña, A., Réthoré, P.-E., & van der Laan, M. P. (2016). On the application of the Jensen wake model using a turbulence-dependent wake decay coefficient: The sexbierum case. *Wind Energy*, 19(4), 763–776. <https://doi.org/10.1002/we.1863>
- Petersson, P., & Krumscheid, S. (2021). Adaptive stratified sampling for non-smooth problems. *arXiv Preprint arXiv*. Retrieved April 4, 2022. <https://doi.org/10.48550/arXiv.2107.01355>
- Pousi, J. P., Virtanen, K., & Virtanen, K. (2013). Simulation metamodeling with Bayesian networks. *Journal of Simulation*, 7(4), 297–311. <https://doi.org/10.1057/jos.2013.18>
- Prudius, A. A., & Andradóttir, S. (2012). Averaging frameworks for simulation optimization with applications to simulated annealing. *Naval Research Logistics (NRL)*, 59 (6), 411–429. <https://doi.org/10.1002/nav.21496>
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1), 81–106. <https://doi.org/10.1007/BF00116251>
- Ross, S. (2013). Chapter 9 - variance reduction techniques. In S. Ross (Ed.), *Simulation* (5th ed, pp. 153–231). Academic Press.
- Santos, C. H. D., De Queiroz, J. A., Leal, F., & Montevechi, J. A. B. (2022). Use of simulation in the industry 4.0 context: Creation of a digital twin to optimise decision making on non-automated process. *Journal of Simulation*, 16(3), 284–297. <https://doi.org/10.1080/17477778.2020.1811172>
- Sargent, R. G. (2010). Verification and validation of simulation models. In *Proceedings of the 2010 winter simulation conference*, Baltimore, MD, USA (pp. 166–183).
- Schruben, L. W. (1980). Establishing the credibility of simulations. *Simulation*, 34(3), 101–105. <https://doi.org/10.1177/003754978003400310>
- Sethi, V. (1963). A note on optimum stratification of populations for estimating the population means. *The Australian Journal of Statistics*, 5(1), 20–33. <https://doi.org/10.1111/j.1467-842X.1963.tb00134.x>
- Shashaani, S., Hashemi, F. S., & Pasupathy, R. (2018). ASTRO-DF: A class of adaptive sampling trust-region algorithms for derivative-free stochastic optimization. *SIAM Journal on Optimization*, 28(4), 3145–3176. <https://doi.org/10.1137/15M1042425>
- Shashaani, S., Hunter, S. R., & Pasupathy, R. (2016). ASTRO-DF: Adaptive sampling trust-region optimization algorithms, heuristics, and numerical experience. In T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, & S. E. Chick (Eds.), *2016 Winter Simulation Conference (WSC)*, Washington, DC, USA (pp. 554–565).
- Singh, R., & Sukhatme, B. (1969). Optimum stratification. *Annals of the Institute of Statistical Mathematics*, 21(1), 515–528. <https://doi.org/10.1007/BF02532275>
- Smith, T. M. (1991). Post-stratification. *Journal of the Royal Statistical Society Series D: The Statistician*, 40(3), 315–323. <https://doi.org/10.2307/2348284>
- Suri, R. (1987). Infinitesimal perturbation analysis for general discrete event systems. *Journal of the ACM (JACM)*, 34(3), 686–717. <https://doi.org/10.1145/28869.28879>
- Taga, Y. (1967). On optimum stratification for the objective variable based on concomitant variables using prior information. *Annals of the Institute of Statistical Mathematics*, 19(1), 101–129. <https://doi.org/10.1007/BF02911670>
- Tahmasebi, F., Zach, R., Schuß, M., & Mahdavi, A. (2012). Simulation model calibration: An optimization-based approach.
- Tipton, E. (2013). Stratified sampling using cluster analysis: A sample selection strategy for improved generalizations from experiments. *Evaluation Review*, 37(2), 109–139. <https://doi.org/10.1177/0193841X13516324>
- Tolk, A., Fowler, J., Shao, G., & Yücesan, E. (Eds.). (2017). Simulation foundations, methods and applications. In *Advances in Modeling and Simulation, Seminal Research from 50 Years of Winter Simulation Conferences* (1st ed., Vol. 10, p. 355). Springer Cham. <https://doi.org/10.1007/978-3-319-64182-9>
- Vasquez, D., Shashaani, S., & Pasupathy, R. (2019). (ASTRO) for derivative-based stochastic optimization: Algorithm description numerical experiments. In C. S. P. H. N. Mustafee, K.-H.-G. Bae, & Y.-J. Son (Eds.), *2019 Winter Simulation Conference (WSC)* National Harbor, MD, USA (pp. 3563–3574).
- Wilson, J. R., Pritsker, A. A. B., & Pritsker, B. (1984). Variance reduction in queueing simulation using generalized concomitant variables. *Journal of Statistical Computation and Simulation*, 19(2), 129–153. <https://doi.org/10.1080/00949658408810723>
- You, M., Byon, E., Jin, J., & Lee, G. (2017a). When wind travels through turbines: A new statistical approach for characterizing heterogeneous wake effects in multi-turbine wind farms. *IIEE Transactions*, 49(1), 84–95. <https://doi.org/10.1080/0740817X.2016.1204489>
- You, M., Liu, B., Byon, E., Huang, S., & Jin, J. (2017b). Direction-dependent power curve modeling for multiple interacting wind turbines. *IEEE Transactions on Power Systems*, 33(2), 1725–1733. <https://doi.org/10.1109/TPWRS.2017.2737529>
- Yuan, J., Ng, S. H., & Tsui, K. L. (2012). Calibration of stochastic computer models using stochastic approximation methods. *IEEE Transactions on Automation Science and Engineering*, 10(1), 171–186. <https://doi.org/10.1109/TASE.2012.2199486>
- Yuan, Y. X. (2015). Recent advances in trust region algorithms. *Mathematical Programming*, 151(1), 249–281. <https://doi.org/10.1007/s10107-015-0893-2>
- Zhao, P., & Zhang, T. (2014). Accelerating minibatch stochastic gradient descent using stratified sampling. *ArXiv Preprint*. Retrieved March 12, 2022, from <https://doi.org/10.48550/arXiv.1405.3080>