Sequential Design and Spatial Modeling for Portfolio Tail Risk Measurement

Jimmy Risk[†]   Michael Ludkovski[*]

May 18, 2018

Abstract

We consider calculation of capital requirements when the underlying economic scenarios are determined by simulatable risk factors. In the respective nested simulation framework, the goal is to estimate portfolio tail risk, quantified via VaR or TVaR of a given collection of future economic scenarios representing factor levels at the risk horizon. Traditionally, evaluating portfolio losses of an outer scenario is done by computing a conditional expectation via inner-level Monte Carlo and is computationally expensive. We introduce several inter-related machine learning techniques to speed up this computation, in particular by properly accounting for the simulation noise. Our main workhorse is an advanced Gaussian Process (GP) regression approach which uses nonparametric spatial modeling to efficiently learn the relationship between the stochastic factors defining scenarios and corresponding portfolio value. Leveraging this emulator, we develop sequential algorithms that adaptively allocate inner simulation budgets to target the quantile region. The GP framework also yields better uncertainty quantification for the resulting VaR / TVaR estimators that reduces bias and variance compared to existing methods. We illustrate the proposed strategies with two case-studies in two and six dimensions.

Keywords: Value-at-Risk estimation, Gaussian process regression, sequential design, nested simulation, portfolio tail risk

1 Introduction

The latest insurance and financial regulations mandate estimation of quantile-based portfolio risk measures. The Solvency II [17] framework calls for computing the 99.5%-level *value-at-risk* (VaR) for a 1-year horizon, while the Basel III regulations in banking [2] require to report the related *tail value-at-risk* (TVaR). In practice, these quantities are calculated by first generating a representative set $\mathcal{Z}$ of future economic scenarios and then evaluating the empirical loss quantile based on $\mathcal{Z}$. However, due to the underlying cashflow and valuation complexity, directly computing future portfolio value is usually not feasible and instead approximations are used. This is done for each

scenario by a Monte Carlo evaluation of the conditional expectation that probabilistically defines portfolio value, leading to a *nested* simulation problem.

In traditional nested simulation, each outer scenario is treated independently, so the respective portfolio loss is approximated through the sample average across the inner simulations that define the resulting cashflows. The scenarios $\mathcal{Z}$ are typically realizations of underlying stochastic factors or risk drivers (Z_t), such as interest rate factors, equity price factors, mortality factors, macroeconomic factors, et cetera. Thus, we can identify each $z^n \in \mathcal{Z}$ as the value of Z_T in scenario ω^n at the risk horizon T; the portfolio value $f(z)$ is identified with the expected cashflow $Y(\cdot)$ (which depends on the future path $(Z_t)_{t \geq T}$ beyond T) conditional on $Z_T = z$:

$$f(z) \doteq \mathbb{E}\left[Y((Z_t)_{t \geq T}) \mid Z_T = z\right]. \tag{1}$$

Note that we assume that Z is Markovian which is essentially always the case in the practical context; if necessary Z is augmented to make it Markov. The more important assumption is that $f(z)$ is not available in closed form, so for a given z it must be approximated via the *inner* step of the nested procedure.

Our goal is to efficiently estimate VaR_α and/or TVaR_α of $f(Z_T)$ when the risk drivers follow a simulatable Markov process. Towards this end we build upon two fundamental strategies that have been developed for nested simulation. The first idea is to improve the inner simulations by adaptively allocating the *simulation budget* to scenarios with larger portfolio losses. Indeed with the mentioned quantile levels, $\approx 99\%$ of the outer scenarios are irrelevant from the point of view of VaR / TVaR computation: only the tail scenarios matter. The second idea is to exploit the *spatial* structure of $\mathcal{Z}$ coming from (Z_t): nearby inputs z, z' should produce similar values $f(z), f(z')$. Therefore, we can improve estimation of $f(z)$ by borrowing information from inner simulations at nearby outer scenarios. This brings a regression perspective that converts the local strategy of computing a sample average into a global goal of approximating the function $z \mapsto f(z)$.

In this article we combine and greatly extend these approaches by embedding them in the framework of *statistical emulation*. Emulation treats f as an unknown function and seeks to produce a functional estimate $\hat{f}$ that optimizes a given objective criterion. For our emulation paradigm we propose *Gaussian process (GP) regression* [31, 34], or *kriging*, which has become a leading choice in the machine learning community. GP's bring a probabilistic (i.e. Bayesian for a statistician) perspective, so that inference of f is viewed as conditioning on the observed simulation output, and estimating a risk measure as computing a posterior expectation of the corresponding random variable. This is arguably the natural framework in the Monte Carlo context where all outputs are intrinsically stochastic. GP's also bring a non-parametric regression approach, alleviating the question of finding good functional approximation spaces for the spatial regression component. Below we show that the (GP) emulation approach, in combination with tailored active learning techniques, can provide accurate estimates even with a small simulation budget, while maintaining a relatively low numerical overhead (fitting, prediction, etc.). Thus, our main message is that significant efficiency can be squeezed from the proposed statistical tools, dramatically improving upon nested simulation.

In the language of emulation, Equation (1) defines a black-box function which we wish to learn. The origins of emulations/GP regression are in analysis of *deterministic* computer experi-

ments where $f(z)$ can be (expensively) evaluated without any noise, see e.g. [18, 22, 28] and the monographs by Santner et al. [34] and Rasmussen and Williams [31]. More recently, emulation was adapted to the Monte Carlo setting where $f(z)$ is only noisily observed: see e.g. [1, 6, 14]. Also related is our earlier work [32] that studied approximate pricing of deferred longevity-linked contracts, requiring an average of Equation (1) over the whole distribution of Z_T.

For tail risk estimation, the learning objective is inherently different from the standard L^2 criterion, since the fitted $\hat{f}$ only needs to be accurate in the tail of $f(Z_T)$. Indeed, as mentioned 99%+ of outer scenarios will turn out to be irrelevant. To our knowledge, there is little literature that directly addresses quantile/tail estimation in the Monte Carlo setting of (1). We mention the work of Oakley [29] who introduced the use of GP's for quantile estimation of deterministic computer experiments and Liu and Staum [27] who pioneered stochastic kriging for estimating TVaR_α in nested Monte Carlo of financial portfolios. Two further important papers are by Bauer et al. [3] who elucidated the use of (parametric linear) regression for tail risk estimation, and Broadie et al. [8] who proved some properties of such Least Squares Monte Carlo methods for learning VaR_α. More broadly, our localized objective resembles three inter-related formulations in the engineering reliability literature that target $f(z)$ in relation to a given level or threshold L:

- Contour-finding: determining the set $\{z : f(z) \in (L - \varepsilon, L + \varepsilon)\}$, ε small, cf. [26, 30];

- Inference of the excursion set $\{z : f(z) \le L\}$ [15];

- Estimating the probability of failure, aka "excursion volume", $\mathbb{P}(\{z : f(z) \le L\})$ [4, 16].

Note that the contour $\{z : f(z) = L\}$ is the boundary of the level set $\{z : f(z) \le L\}$, and the excursion volume is a weighted integral over the level set. However, in all of the above (except [26]), the threshold L is exogenously given, while in the Value-at-Risk context, the desired quantile VaR_α is implicitly defined via f itself, see (3) below. Moreover, all of [4, 15, 26, 30] dealt with deterministic noise-free experiments.

At the heart of emulation is design of experiments, i.e. determining which simulations to run so as to *learn* the input-output pairing $z \mapsto f(z)$ as quickly as possible. This naturally connects to adaptive allocation of inner simulations. Adaptivity is achieved by dividing the overall simulation into several *stages* that gradually learn the shape of f. The initial stage uses a fraction of the simulation budget to learn about $f(z)$ on the whole domain. The resulting search region targeting the objective is then refined through further stages. For example, the approach in [29] has two stages: in the second stage inner simulations are non-uniformly allocated to scenarios in the quantile region based on the initial inference about f. Liu and Staum [27] proposed a total of three stages; during the second stage another fraction of the budget is allocated uniformly to scenarios in the estimated tail, and finally the remaining budget is dispersed (non-uniformly) among the tail scenarios to minimize the posterior variance of the TVaR_α estimator.

Further, inspired by optimization objectives (finding the global maximum of f), there are *sequential design* approaches which use a large number of stages (in principle adding just 1 simulation per stage). In particular, *stepwise uncertainty reduction* (SUR) methods define an acquisition function $\mathcal{H}(\cdot)$ and greedily pick scenarios z that maximize $\mathcal{H}(z)$ conditional on current data $\mathcal{D}$. For example, Picheny et al. [30] sequentially sample f at scenarios according to a targeted expected

improvement criterion. Another type of adaptive strategies does not rely on emulation/spatial modeling, rather utilizing the tools of what is known as ranking-and-selection (R&S) [12, 19]. However, R&S procedures are generally based on doing a hypothesis test for $f(z^n) > L$ in parallel for each scenario z^n. Theoretically this allows to decouple the estimation problems, but such schemes are impractical if L is itself unknown.

In the VaR context, several aspects of the setting make the emulation problem statistically challenging. First, the typical number of outer scenarios N is quite large, on the order of $N \in [10^4, 10^5]$ for practitioners. This renders standard emulation and R&S strategies computationally heavy. Furthermore, the typical simulation budget $\mathcal{N}$ is often quite small, $\mathcal{N} \sim N$, which means that a brute-force uniform allocation approach can only sample each scenario a handful of times, leading to disastrous estimates. It further implies that practical allocation schemes must be quite *aggressive* and the asymptotic guarantees available in the R&S literature are not applicable. At the same time $\mathcal{N}$ is large by emulation standards which treat evaluations of f as extremely expensive (typical budget is $\ll 500$ simulations) and hence the associated algorithms often carry unacceptable computational overhead. Second, the outer scenarios are usually treated by practitioners as a fixed object (namely coming from an exogenous Economic Scenario Generator aka ESG). In other words, the scenario space $\mathcal{Z}$ is taken to be discrete; one cannot add (or subtract) further scenarios like in the strategy of Broadie et al. [8]. Neither can one employ continuous search methods that are popular in the simulation optimization literature. Third, the nature of the underlying cashflows makes the simulation noise in the cashflows Y highly non-Gaussian. It is typically skewed (because many cashflows have embedded optionality and hence the corresponding distribution has point masses) and with low signal-to-noise ratio. The latter implies that cross-scenario information borrowing is crucial to maximize accuracy. Fourth, the portfolio losses are highly inhomogeneous, i.e. ϵ is *heteroskedastic* which strongly affects the inner simulation allocations. Fifth, because the objective function is implicitly defined in terms of f, constructing an appropriate estimator, and especially quantifying its accuracy (aka standard error) is nontrivial in its own right.

The framework that we propose overcomes all of the above challenges and consists of three key pieces. First, we connect to the burgeoning literature on level-set estimation for deterministic computer experiments, tailoring the recent successes of the SUR techniques [16] to the VaR / TVaR problem. As far as we know this is the first link between these disparate literatures, which we see as an invitation towards closer marriage of machine learning and simulation-based risk measurement. Second, we work with a state-of-the-art GP emulator. By their nature, GPs offer rich uncertainty quantification properties, in particular many analytic formulas for active learning criteria that are used in guiding the simulation allocation. We take this a step further, employing an advanced GP methodology [5, 6] that is customized for the Monte Carlo simulation context. Third, we take advantage of the *discrete scenario set* which intrinsically calls for a replicated design. In turn, replication yields (i) improved noise properties that minimize non-Gaussianity; (ii) ability to simultaneously learn the mean response $f(\cdot)$ and the conditional simulation variance $\tau^2(\cdot)$ to handle heteroskedasticity; (iii) reduced model overhead; (iv) convenient GP implementation. The symbiotic relationship of replication and kriging was already observed in [1]; the `hetGP` [5] library that we use gracefully handles all aspects of (i)-(iv).

Rather than proposing a single specialized algorithm, we present a general framework comprising

several modules. By adjusting these "moving parts", the algorithm can (a) evaluate different risk-measures (we illustrate with both VaR and TVaR); (b) use different emulator codes; (c) switch between different ways of running the sequential budget allocation such as the initialization phase; number of sequential stages; termination criterion; batch-size and so on; (d) rely on different VaR estimators. To illustrate the above choices we present extensive numerical illustrations that compare the impact of different modules. In two case studies, we compare it with benchmarks, such as the algorithm in Liu and Staum [27].

Our statistical learning perspective implies a preference for *sequential* algorithms that employ the feedback loop of simulate–estimate–assess–simulate–... Compared to classical one-/two-stage designs, the fully sequential strategies offer several attractive features. First, they internalize the uncertainty quantification that is offered by the emulator and therefore lead to a more automated "artificial intelligence" implementation. Second, they can be run online, i.e. on a server or in the cloud with interim results always available for inspection, while the algorithm proceeds to refine its accuracy. Thus, the user no longer has to specify the simulation budget (i.e. number of inner simulations one is willing to run) a priori, but instead interacts with the Monte Carlo on-the-fly. Third, sequential methods are a starting point for more general re-use of simulations, for example for periodic re-computation of the whole problem as business time progresses (usually VaR calculations are done on a weekly or monthly cycle).

Our overall contribution is to remedy limitations of current VaR/TVaR estimation algorithms through ideas from GP emulators and related sequential design problems. In the context of TVaR, arguably the closest approach, and the one that we are to a large extent indebted to, is the algorithm in Liu and Staum [27]. Relative to [27] we may list the following 4 important improvements. First, we generalize their 3-stage strategy to a fully sequential k-stage procedure, dubbed SV-GP below. In particular, we eliminate the need for their intermediate second stage which was performing a conservative "exploration" (and required finetuning), and also extend their algorithm to handle non-constant replication amounts. Second, we employ the latest `hetGP` emulators which significantly improve the emulation in low-budget environments, in particular by lowering the bias in learning $f(x)$. Third, we provide an alternative class of allocation strategies relying on Active Learning heuristics. Finally, while [27] only treated TVaR, we work (and compare the implementation nuances) with both VaR and TVaR.

The paper is organized as follows: in Section 2 we discuss the emulation objective and portfolio tail risk, as well as VaR_α estimators. Section 3 formally introduces the GP model and its mathematical details. Next, Section 4 develops the sequential design criteria for our problem. Section 5 provides the full implementation. Finally, we apply the algorithm to two case studies: Section 6 involves a two-factor model, where we compare sequential VaR / TVaR strategies to several benchmarks, along with a look at various GP versions; Section 7 investigates performance in higher dimensions where the state process is six-dimensional and the simulator is much more expensive.

2 Objective

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, we consider a stochastic system with Markov state process $Z = (Z_t)$. Both continuous- and discrete-time models can be treated. Typically, Z is a multivariate

stochastic process based on either a stochastic differential equation ($t \in \mathbb{R}_+$) or time-series ARIMA framework ($t \in \mathbb{N}$). Based on the realizations of Z, the modeler needs to assess the resulting capital $f(Z_T)$ at an intermediate horizon T. The portfolio value $f(Z_T)$ is computed as the conditional expected value of discounted cashflows Y_t given $\mathcal{F}_T$, the information at time T. The cashflow $Y_t(z.)$ at date t could depend on the whole path $Z_{[T,t]}$ of the stochastic factor from T to t (but not on $z_{[0,T]}$). To compute their net present value Y, we discount at a constant risk-free rate β

$$Y \doteq \sum_{t=T}^{\infty} e^{-\beta(t-T)} Y_t(Z_{[T,t]}). \tag{2}$$

Since Z is Markov, we can write the conditional expectation of Y given $\mathcal{F}_T$ as a function $f(z)$ defined in (1), and similarly will henceforth use the notation $Y(z)$. We assume that $f(\cdot)$ does not possess any simple/closed-form functional form. However, since $f(z)$ is a conditional expectation, it can be sampled using a simulator, i.e. the modeler has access to an engine that can generate independent, identically distributed trajectories of (Z_t), $t \geq T$.

Fixing a quantile α (most commonly $\alpha = 0.005$ corresponding to 99.5%-VaR level mentioned in the Introduction), the *capital requirements* based on VaR_α and TVaR_α are respectively the threshold such that the probability of loss exceeding VaR_α is[1] α, i.e.

$$\mathrm{VaR}_\alpha(f(Z_T)) \doteq \sup\{x : \mathbb{P}(f(Z_T) \leq x) \leq \alpha\}, \tag{3}$$

and the $\alpha-$tail average,

$$\mathrm{TVaR}_\alpha(f(Z_T)) \doteq \mathbb{E}\big[f(Z_T) \mid f(Z_T) \leq \mathrm{VaR}_\alpha(f(Z_T))\big]. \tag{4}$$

As mentioned, rather than working with the theoretical distribution of Z_T, we consider a given scenario set $\mathcal{Z}$ of size N which is assumed to be fixed for the remainder of the presentation. Typically the outer scenarios are provided by an Economic Scenario Generator (ESG) that calibrates the dynamics of (Z_t) to the historical data, i.e. it is under the physical measure $\mathbb{P}$. The inner simulations are based on a mark-to-market law, i.e. it is under the risk-neutral measure $\mathbb{Q}$. Therefore, the evolution of (Z_t) is different on $[0, T]$ and on $[T, \infty)$. This is one reason for treating $\mathcal{Z}$ as fixed, allowing us to eschew discussion of the precise mechanism of the ESG/outer simulator.

Given the loss scenarios $\{z^1, z^2, \ldots, z^N\} \doteq \mathcal{Z}$, computing VaR_a now translates into finding the αN (assumed to be integer) order statistic $f^{(\alpha N)}$ of $f^{1:N}$, where we use the shorthand $f^n \doteq f(z^n)$, and the superscripts (n) refer to the nth ordered value (in increasing order) of $f^{1:N}$. More generally, we consider risk measures R for $\mathcal{Z}$ defined through weights: given a collection of losses $f^{1:N}$ let

$$R \doteq \sum_{n=1}^{N} w^n f^n, \tag{5}$$

where $\sum_{n=1}^{N} w^n = 1$. For VaR_α and TVaR_α the weights are respectively

$$w^{n,\mathrm{VaR}} = \mathbb{1}_{\{n:f^n=f^{(\alpha N)}\}}; \tag{6}$$

$$w^{n,\mathrm{TVaR}} = \frac{1}{\alpha N} \cdot \mathbb{1}_{\{n:f^n \leq f^{(\alpha N)}\}}. \tag{7}$$

---

[1]Throughout we view f as portfolio value, i.e. more is better and the tail risk concerns the left tail where f is most negative.

Another way to think about the tail risk is through the corresponding *tail regions*

$$\mathfrak{R}^{\text{VaR}} \doteq \{z^n : f^n = f^{(\alpha N)}\}, \qquad \text{and} \quad \mathfrak{R}^{\text{TVaR}} \doteq \{z^n : f^n \leq f^{(\alpha N)}\}. \tag{8}$$

Finally, we also introduce the notation $\mathfrak{q}_\alpha$ to denote the exact α-quantile scenario (i.e. the one in $\mathfrak{R}^{\text{VaR}}$): $f(\mathfrak{q}_\alpha) = f^{(\alpha N)}$.

2.1 Tail Risk Estimation with Nested Simulation

The standard Monte Carlo approach is based on nested simulation. Starting from $\mathcal{Z}$, for each n, $f(z^n)$ is approximated by an inner empirical average

$$f(z^n) \simeq \bar{y}^n \doteq \frac{1}{r^n} \sum_{i=1}^{r^n} y^{n,i}, \qquad n = 1, \dots, N, \tag{9}$$

where $y^{n,i}$ is the present value of loss from (2) based on independent replications $Y^{n,i}_\cdot(z^n_\cdot)$, $i \in \{1, \dots, r^n\}$ evaluated through the trajectories $z^{n,i}_\cdot$ of $(Z_t)_{t \geq T}|Z_T = z^n$. Equations (3) and (4) are then estimated by a sample quantile and tail average respectively. Specifically, we sort the estimated scenario losses $(\bar{y}^n)_{n=1}^N$ in increasing order $\bar{y}^{(1)} \leq \bar{y}^{(2)} \leq \dots \leq \bar{y}^{(N)}$ and take $\hat{R}^{SA,\text{VaR}} \doteq \bar{y}^{(\alpha N)}$ and $\hat{R}^{SA,\text{TVaR}} \doteq \frac{1}{\alpha N} \sum_{i=1}^{\alpha N} \bar{y}^{(i)}$.

In the simplest setting, the inner budget is constant across scenarios, $r^n = \bar{r}$ leading to total simulation budget of $\mathcal{O}(N \cdot \bar{r})$. This quickly becomes expensive: for example, a budget of $\bar{r} = 10^3$ and $N = 10^3$ scenarios requires 10^6 total simulations. Moreover, the empirical estimator $\hat{R}^{SA,\text{VaR}}$ is biased, see Kim and Hardy [25] (who point out that one cannot quantify which direction the bias lies without knowing some details about f and the distribution of Z_T) and Gordy and Juneja [21]. These papers discuss ways of reducing bias, through bootstrap and jackknife, respectively. Gordy and Juneja [21] also provide consequent optimal strategies for choosing N and $\bar{r}$ for a fixed budget $N\bar{r} = \mathcal{N}$.

As another way to alleviate the bias in $\hat{R}^{SA,\text{VaR}}$, one may construct several modified versions of VaR_α estimators that stem from weighted averages of nearby order statistics. Called $L-$estimators [36], they offer robustness compared to a single sample order statistic. Effectively, such estimators modify the weights w^n defining (5) to account for the uncertainty in the marginal estimators $\hat{f}(z^n)$. A well known construction is by Harrell and Davis [23], where the weights for a VaR_α estimator are chosen as $R^{HD,\text{VaR}} = \sum_n \tilde{w}^{(n)} \hat{f}^{(n)}$ with

$$\tilde{w}^{(n)} = \int_{(n-1)/N}^{n/N} \text{Beta}(t; (N+1)\alpha N, (N+1)(1 - \alpha N)) dt, \tag{10}$$

where $\text{Beta}(x; a, b)$ is the Beta-distribution density function with shape parameters (a, b) and the index (n) refers to the order statistics of $\hat{f}$ (e.g. the $\bar{y}^n, n = 1, \dots, N$ if one uses the empirical plug-in approach). The weights $\tilde{w}^{(n)}$ are largest near $n = \alpha N$ and taper quickly on either side, smoothing out the binary 0/1 nature of (6). We refer to [35, 36] for further details about the Harrell-Davis weights and their performance relative to other $L-$estimators. While the general conclusion is that no one estimator performs best, in our experiments we find a clear preference for (10) over the

original (6) both in terms of smaller bias and lower standard error. These concerns do not extend to the TVaR case, since in our experience the empirical estimator $\hat{R}_\alpha^{\text{TVaR}} = \frac{1}{\alpha N} \sum_{n=1}^{\alpha N} \hat{f}^{(n)}$ is already robust enough against mis-specification of a few borderline scenarios. Theoretically, additional robustness could be achieved via smoothing the cut-off in $\hat{w}^{n,\text{TVaR}}$ near $\hat{f}^{(\alpha N)}$ to create a structure similar to (10).

3 Gaussian Process Emulation

To construct thrifty schemes for approximating (3) and (4), we replace the inner step of repeatedly evaluating $f(z)$ by a *surrogate model* $\hat{f}$ for f. This emulation framework generates a fitted $\hat{f}$ by solving regression equations over a training dataset. In contrast to classical regression, the modeler is in charge of the experimental design (building the training samples) which are viewed as arriving sequentially. Particularly for tail risk, we seek a procedure that identifies the tail scenarios that matter for (3)-(4). We note that emulation offers a statistical perspective on approximating f. An alternative, commonly employed by practitioners, is to construct analytic approximations, for instance by integrating out some of the randomness in $Z_{[T,\infty)}$ to obtain a closed-form expression for $f(z)$. However, the quality of such formulas is hard to judge and they often require customized derivations. In contradistinction, the surrogate $\hat{f}$ directly smoothes out the Monte Carlo noise from nearby scenarios and is constructed to be asymptotically consistent with the true f as the budget increases.

Formally, the statistical problem of emulation deals with a sampler

$$Y(z) = f(z) + \epsilon(z), \tag{11}$$

where we identify $f(\cdot)$ with the unknown *response surface* and $\epsilon(\cdot)$ is the sampling noise, with variance $\tau^2(z)$, assumed to be independent and identically distributed across different calls to the sampler. Specifically, the noise is taken to be Gaussian $\epsilon(z) \sim \mathcal{N}(0, \tau^2(z))$, but scenario-dependent. Emulation now involves the (i) experimental design step of proposing a design $\mathcal{D}_k$ at each stage k of the procedure that forms the training dataset, and (ii) a learning procedure that uses the collected outputs $\mathcal{D}_k = (z^n, \mathbf{y}_k^n)_{n=1}^N$, with $\mathbf{y}_k^n = \{y_k^{n,1}, \ldots, y_k^{n,r_k^n}\}$ being a collection of r_k^n realizations of (11) at z^n, to construct a fitted response surface $\hat{f}(\cdot)$ which is based on the conditional law of f over the appropriate functional space, $\mathcal{L}(f|\mathcal{D}_k)$. Here, we consider the case where $\hat{f}$ is fitted sequentially based on a multi-step procedure, and the subscript k denotes the step counter, so that $\mathcal{D}_k \subseteq \mathcal{D}_{k+1}$.

3.1 GP Posterior

A GP surrogate assumes that f in (11) has the form

$$f(z) = \mu(z) + X(z), \tag{12}$$

where $\mu : \mathbb{R}^d \to \mathbb{R}$ is a trend function, and X is a mean-zero square-integrable Gaussian process with covariance kernel $C(x, x')$. The role of C is to generate the reproducing kernel Hilbert space $\mathcal{H}_C$ which is the functional space that X is assumed to belong to.

8

Since the noise $\epsilon(z)$ is also Gaussian, we have that $\hat{f}_k(z) \equiv f(z)|\mathcal{D}_k \sim \mathcal{N}(\mu(z) + m(z), s^2(z))$ has a Gaussian posterior, which reduces to computing the kriging mean $m(z)$ and kriging variance $s^2(z)$. One can then take $\mu(z) + m(z)$ as the point estimate of f at scenario z. In turn, the kriging variance $s^2(z)$ offers a principled empirical estimate of model accuracy, quantifying the approximation quality. In particular, one can use $s^2(z)$ as the proxy for the mean-squared error (MSE) of $\hat{f}$ at z. The Gaussian process property implies that not only is the marginal posterior at z Gaussian, but also the joint distribution of $(f(z'^1), \dots, f(z'^M))|\mathcal{D}_k$ is multivariate normal (MVN) for any M-tuple $z'^{1:M}$.

By considering the process $f(z) - \mu(z)$, we may assume without loss of generality that f is statistically centered at zero. Denoting the sample average at each scenario z^n by $\overline{y}_k^n = \frac{1}{r_k^n} \sum_{i=1}^{r_k^n} y_k^{n,i}$ as in Equation (9) and the overall collection as $\overline{\mathbf{y}}_k \doteq (\overline{y}_k^1, \dots, \overline{y}_k^N)$ (note that here we assume that all outer scenarios have been already sampled, see further discussion on p. 14), the resulting posterior mean and variance of $\hat{f}_k(z)$ follow from the MVN conditional equations [33]

$$\begin{cases} m_k(z) \doteq \mathbf{c}(z)(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\overline{\mathbf{y}}_k^T; \\ s_k^2(z) \doteq C(z, z) - \mathbf{c}(z)(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{c}(z)^T, \end{cases} \tag{13}$$

where $\mathbf{c}(z) = \left(C(z, z^j) \right)_{1 \le j \le N}$, $\mathbf{C} \doteq \left[C(z^i, z^j) \right]_{1 \le i,j \le N}$, $\boldsymbol{\Delta}_k$ is the $N \times N$ diagonal matrix with entries $\tau^2(z^1)/r_k^1, \dots, \tau^2(z^N)/r_k^N$ and ^{T} denotes transpose. Note that the conditional variance $\tau^2(\cdot)$ is typically unknown, so applying (13) requires replacing it with a further approximation $\hat{\tau}^2(\cdot)$, as discussed in Section 3.3.

Remark. For readers unfamiliar with emulators and surrogates, the idea is equivalent to regression: projecting the unknown f onto a specified approximation space $\mathcal{H}$. The latter is "non-parametric" in the sense of being infinite-dimensional and dense in the class of continuous functions, so that for an input dataset of size N, there are $\mathcal{O}(N)$ degrees of freedom. In contrast to classical regression, in emulation there is no "data": the controller is also in charge of proposing scenarios x where f should be probed; because the outputs are stochastic it makes perfect sense to batch, i.e. probe the same scenario multiple times. GP emulation treats f as a realization of a random field, recasting regression as probabilistically conditioning this random field on the given input/output pairs. This is equivalent to taking $\mathcal{H}$ to be the RKHS generated by kernel C and then applying the standard L^2-projection equations.

3.2 Covariance kernels and hyperparameter estimation

Fitting a GP emulator is equivalent to specifying the covariance function $C(\cdot, \cdot)$ and applying (13). The standard choice is a spatially stationary kernel,

$$C(z, z') \equiv c(z - z') = \sigma^2 \prod_{j=1}^{d} g(z_j - z'_j; \theta_j),$$

reducing to the one-dimensional base kernel g. Below we use the Matérn-5/2 kernel

$$g(h; \theta) = \left(1 + \frac{\sqrt{5}h}{\theta} + \frac{5h^2}{3\theta^2} \right) \exp\left(-\frac{\sqrt{5}h}{\theta} \right). \tag{14}$$

The hyper-parameters θ_j are called characteristic length-scales and informally correspond to the distance between inputs over which the response f can change significantly [31, Ch 2]. We utilize the R packages `DiceKriging` [33] and `hetGP` [5] that fit the hyper-parameters σ, θ_j by Maximum Likelihood. The latter is a nonlinear optimization problem and the packages differ in how the search for the global maximum (initialization, gradient estimation, etc.) is conducted.

3.3 Intrinsic Variance

The simulation variance $\tau^2(z)$ is scenario-dependent and a crucial piece of fitting a surrogate. In a basic GP emulator, it is approximated by a local empirical variance, leveraging the replications offered by multiple inner simulations. This approach, known as Stochastic Kriging (SK) and introduced in Ankenman et al. [1], sets

$$\hat{\tau}_k^2(z^n) \doteq \frac{1}{r_k^n - 1} \sum_{i=1}^{r_k^n} (y_k^{n,i} - \bar{y}_k^n)^2. \tag{15}$$

However, SK requires $r_k^n \geq 2$ and more realistically $r_n^k \gg 10$, as an insufficient number of inner simulations will give increasingly unreliable estimates of the conditional variance. This places a major restriction on the sequential algorithms which are then required to sample all relevant scenarios a minimal number of times. Moreover, $\hat{\tau}$ in (15) is only available a posteriori, so there is no natural way to predict simulation variance at a yet-to-be-sampled scenario (Ankenman et al. [1] suggested to do a GP-based interpolation of $\hat{\tau}^2$ for that purpose).

To overcome this limitation, we use an extended GP model [6] that jointly models the response mean $z \mapsto f(z)$ and (log)-variance $z \mapsto \log \tau^2(z)$. Specifically, an additional latent variable Λ_n is introduced for each scenario $z^n \in \mathcal{D}$. The estimated τ^2's are then viewed as *spatially smoothed* versions of Λ_n, combining the above interpolation idea of SK with additional smoothing to account for the uncertainty about the true $\tau^2(z^n)$. The model jointly learns the hyperparameters defining the covariance kernel $C(\cdot, \cdot)$ of $f(\cdot)$, and all the Λ_n's. While this increases the complexity of the underlying optimization problem that is solved during the likelihood maximization step, tractability is maintained thanks to analytic expressions for the respective likelihood function gradients. We refer to Binois et al. [6] for the full details and utilize the resulting `hetGP` [5] package where this procedure is efficiently implemented. Relative to SK, the `hetGP` approach has no lower-bound constraint on r_k^n's and is especially beneficial at the early stages of the algorithm where many of the $\hat{\tau}$'s are unstable. According to [6], using `hetGP` improves performance even for homoskedastic models, simply through better handling of the replication, avoiding the pitfalls of (15). In our experiments, `hetGP` performed much better, in particular lowering the bias in $m_k(\cdot)$ for small k, and hence improving the "zooming in" feature of our sequential design strategies.

3.4 Estimating Portfolio Risk

After a surrogate for $f(\cdot)$ is constructed, there remains the problem of estimating the risk measure R defined in (5). This *identification* problem is itself non-trivial, not least because we seek more than a point estimate $\hat{R}$.

10

Taking a probabilistic point of view, R as a random variable (defined through the random variable $f^{1:N}$) that carries its own posterior distribution given $\mathcal{D}_k$. Thus, a GP-based estimate of R is

$$\hat{R}_k^{GP} \doteq \mathbb{E}[R|\mathcal{D}_k] = \sum_n \mathbb{E}[w^n f(z^n)|\mathcal{D}_k]. \tag{16}$$

Because the weights w^n are defined implicitly in terms of order statistics of $f(z^{1:N})$, $\hat{R}^{GP}$ requires integrating against the joint distribution of $\hat{f}^{1:N}$. While the latter is multivariate Gaussian, there are no closed-form formulas for the probability that one coordinate $\hat{f}^n$ of a MVN distribution is a particular order statistic $\hat{f}^{(\alpha N)}$ (though see [26] who develop a related approximation). Nevertheless, the conditional expectation in (16) can be in principle numerically approximated by making draws from the posterior MVN law of $\hat{f}^{1:N}$, evaluating the resulting quantile/tail and averaging.

A much cheaper solution is to work with the kriging means $m_k(z)$ in analogue to the standard plug-in estimators $\hat{R}^{SA}$ based on $\bar{y}$'s in nested simulation. Specifically, we use the Harrell-Davis $L-$estimator (10) based on the sorted posterior means:

$$\hat{R}_k^{HD,\mathrm{VaR}} \doteq \sum_n \tilde{w}^{(n)} m_k^{(n)}. \tag{17}$$

For estimating TVaR_α, the empirical weights $\hat{w}_k^{n,\mathrm{TVaR}} = \frac{1}{\alpha N}\mathbb{1}_{\{m_k(z^n) \leq m_k^{(\alpha N)}\}}$ based on the posterior means are used analogously to (17).

Given an estimator $\sum_n \hat{w}_k^{(n)} \hat{f}_k^{(n)}$ and denoting $\hat{\mathbf{w}}_k \doteq (\hat{w}_k^1, \ldots, \hat{w}_k^N)$, its mean is obtained by plugging in the posterior means m_k^n for $\hat{f}^n$ like in (17) and we can similarly obtain the estimator variance:

$$s^2(\hat{R}_k) \doteq \mathbb{V}\mathrm{ar}\left(\sum_n \hat{w}^{(n)} f_k^{(n)}\Big|\mathcal{D}_k\right) = \hat{\mathbf{w}}_k \left[\mathbf{C} - \mathbf{c}(z)(\mathbf{C}+\boldsymbol{\Delta}_k)^{-1}\mathbf{c}(z)^T\right]\hat{\mathbf{w}}_k^T. \tag{18}$$

Note that the inside term in Equation (18) is the posterior covariance matrix of $\hat{f}_k^{1:N}$, taking advantage of the covariance structure of the GP for additional smoothing. For later use we also define the estimated quantile scenario $\widehat{\mathfrak{q}_\alpha}$ which solves $m_k(\widehat{\mathfrak{q}_\alpha}) = m_k^{(\alpha N)}$ and may be compared to the true quantile $\mathfrak{q}_\alpha$ based on f.

Remark. To motivate the use of $\hat{R}^{HD,\mathrm{VaR}}$, consider the "in-between" approach of starting with (16) but treating the two terms as independent:

$$\mathbb{E}[R|\mathcal{D}_k] \simeq \sum_n \mathbb{E}[f(z^n)|\mathcal{D}_k] \cdot \mathbb{E}[w^n|\mathcal{D}_k] = \sum_n \omega_k^n m_k(z^n),$$

where the weights are (up to a normalizing factor) $\omega_k^n = \mathbb{P}(z^n \in \mathfrak{R}|\mathcal{D}_k)$. This means that we should replace the 0/1 weights in (6)-(7) with their smoothed posterior probabilities. Simulating ω^n in our case studies and plotting them against the ordered means $m^{1:N}$ gives a bell-shape that reasonably matches the Beta shape of the Harrell-Davis weights $\tilde{w}^{(n)}$ in (10). (However note that as $k \to \infty$, $\omega_k^n \to w^{n,Var}$ while the Harrell-Davis weights $\tilde{w}$ are independent of k.)

4 Sequential Design for Tail Approximation

To estimate R we assume being given a total budget of $\mathcal{N}$ inner simulations to allocate across N fixed scenarios. We then apply the following general procedure to split the computation into K sequential rounds $k = 1, \ldots, K$, where $\mathcal{N}_k$ is the *remaining* simulation budget before round k (with $\mathcal{N}_0 \equiv \mathcal{N}$):

I. Initialize $\hat{f}_0$ by generating simulations over a subset of *pilot* scenarios.

II. Sequentially over $k = 0, 1, \ldots$, until $\mathcal{N}_k = 0$, predict $\hat{f}_k$ on $\mathcal{Z}$ to determine which scenarios are close to $\mathfrak{R}$. Allocate more inner simulations to this/these scenario(s), i.e. increase the corresponding r^n's. This is achieved via an *acquisition function* $\mathcal{H}(z^n)$ that takes into account the "closeness" of z^n to $\mathfrak{R}$ and the uncertainty $s_k(z^n)$. Potentially a new outer scenario that previously had $r_k^n = 0$ might be sampled. Then update to produce $\hat{f}_{k+1}$ based on the new MC output.

III. The final estimate $\hat{R}_K$ is obtained from Equation (17), with uncertainty expressed via (18).

Making the above mathematically precise boils down to two objectives: (i) discover the region of $\mathcal{Z}$ corresponding to $\mathfrak{R}$, and (ii) reduce $s_k^2(\cdot)$ in this region. These objectives match the exploration-exploitation tradeoff: allocating too many replications to solve (i) produces a surrogate that lacks precision even though it recognizes the location of $\mathfrak{R}$, while focusing only on (ii) without sufficient searching may zero in on the wrong region. To guide this tradeoff during sequential allocation, the acquisition function is based on an uncertainty measure. The strategy is then to (myopically) carry out *stepwise uncertainty reduction* (SUR, see [4, 14, 15, 29, 30]) by determining what new simulations would most reduce expected uncertainty for the *next* round. Despite their proliferation in the machine learning literature, to our knowledge, we are the first to apply these concepts for portfolio risk measurement.

The computational overhead associated with the loop in Step II is non-negligible. It concerns the computation of the acquisition function $\mathcal{H}_k(z^{1:N})$, as well as the updating of the GP surrogates $\hat{f}_k \to \hat{f}_{k+1}$. For computational efficiency, rather than adding a single inner simulation, we therefore work with batches of size Δr_k, meaning that $\mathcal{N}_{k+1} = \mathcal{N}_k - \Delta r_k$. The sequential design procedure is therefore to determine the best allocation $r_k'^n$ satisfying $\sum_n r_k'^n = \Delta r_k$. The approaches in Sections 4.1-4.2 allocate all Δr_k new replications to a single z^{k+1} scenario. Alternatively the approach in Section 4.3 solves an additional optimization problem on how to distribute the new inner simulations across multiple outer scenarios. For ease of presentation we focus on constant budget per step Δr; for example, $\Delta r = 0.01\mathcal{N}_1$ (where $\mathcal{N}_1$ is the remaining budget after initialization) yields a procedure with $K = 100$ rounds. Further speed-ups to reduce the overhead are discussed in Section 5.3.

Remark. Not all outer scenarios will typically be sampled by our algorithms, especially in the early rounds. Therefore, we distinguish between the total N vs. $N'(k) \leq N$ which is the number of scenarios sampled by round k. Thus, in expressions such as (13) or (17), the effective computation is with matrices/vectors of size $N'(k)$. This is important for computational overhead since a major bottleneck is handling the $N'(k) \times N'(k)$ matrix $\mathbf{C}$.

4.1 Targeted MSE Criterion

For the quantile objective which corresponds to Value-at-Risk computation, the region of interest $\mathfrak{R}^{\text{VaR}}$ is the contour $\{z : f(z) = L\}$, where the level $L = f^{(\alpha N)}$ is implicit. To learn the contour, we wish to reduce the kriging variance for scenarios close to L. To this end we consider the *targeted mean square error* for a scenario z originally introduced in Picheny et al. [30]:

$$\text{tmse}_k^{\text{VaR}}(z) \doteq s_k^2(z) W_k^{\text{VaR}}(z; L) \qquad \text{where}$$

$$W_k^{\text{VaR}}(z; L) \doteq \frac{1}{\sqrt{2\pi(s_k^2(z) + \varepsilon^2)}} \exp\left(-\frac{1}{2}\left(\frac{m_k(z) - L}{\sqrt{s_k^2(z) + \varepsilon^2}}\right)^2\right) = \phi(m_k(z) - L, s_k^2(z) + \varepsilon^2) \quad (19)$$

is based on the Gaussian pdf ϕ. Thus, $W_k^{\text{VaR}}(z; L)$ is largest for scenarios where predicted portfolio value is close to L and scenarios that have higher posterior variance. The parameter ε in (19) controls how localized is the criterion around the level L. Picheny et al. [30] recommended ε to be five percent of the response range, however, they considered noiseless samplers with $\tau^2 \equiv 0$. In our case, it is desirable to have ε decrease as k increases, to reflect improving knowledge about R. We propose to take $\varepsilon_k \doteq s(\hat{R}_k^{HD})$ from (18) which captures the uncertainty about the quantile. This is large when k is small (uncertain in earlier stages), and decreases rapidly as k increases.

The overall acquisition function to be greedily minimized is then $\mathcal{H}_{k+1}^{timse}$, the total (integrated) tmse over all of $\mathcal{Z}$ conditional on adding simulations at z^{k+1}:

$$\mathcal{H}_{k+1}^{timse} \doteq \frac{1}{N} \sum_{n=1}^{N} \text{tmse}_{k+1}^{\text{VaR}}(z^n) = \frac{1}{N} \sum_{n=1}^{N} s_{k+1}^2(z^n) W_{k+1}^{\text{VaR}}(z^n; R). \tag{20}$$

The right hand side of (20) still contains terms that will only be known after sampling at z^{k+1}. Let us define

$$V_k(z^n; z^m) \doteq C(z^n, z^n) - \mathbf{c}(z^n)(\mathbf{C} + \mathbf{\Delta}_{k+1}^{cand})^{-1}\mathbf{c}(z^n)^T \Big|_{\mathbf{\Delta}_{k+1}^{cand} = \text{diag}\left(\frac{\hat{\tau}_k^2(z^1)}{r_k^1}, \dots, \frac{\hat{\tau}_k^2(z^m)}{r_k^m + \Delta r_k}, \dots, \frac{\hat{\tau}_k^2(z^{N'})}{r_k^{N'}}\right)} \tag{21}$$

which approximates the next-step kriging variance at z^n under the assumption that Δr_k additional replications were added to z^m (keeping all other GP pieces frozen from the k-th round). Note that the noise matrix $\mathbf{\Delta}_{k+1}^{cand}$ is affected by the Δr_k's (and might have one more row/column relative to $\mathbf{\Delta}_k$ if a hitherto unsampled scenario z^m is considered), but the covariance matrix $\mathbf{C}$ is not.

We furthermore approximate $W_{k+1}^{\text{VaR}}(z^n; R)$ with the current tmse weight $W_k^{\text{VaR}}(z^n)$ evaluated at the level $\hat{R}_k^{HD}$, so the final criterion is

$$\widehat{\mathcal{H}}_k^{\text{VaR,timse}}(z) \doteq \frac{1}{N} \sum_{n=1}^{N} V_k(z^n; z) W_k^{\text{VaR}}(z^n; \hat{R}_k^{HD}), \tag{22}$$

which is numerically minimized over the next sampling scenario z. Selecting z^{k+1} as the minimizer of (22) is termed the ST-GP (for "Sequential TIMSE based on a GP") procedure.

Remark. The original [30] considered the case where $\mathcal{Z}$ is continuous, so the designs were augmented with new sites z^{k+1} and $\mathcal{H}^{\text{timse}}$ was defined via an integral. In our case $\mathcal{Z}$ is finite ($\mathcal{H}^{\text{timse}}$ is a sum) and fixed, so we add *replications* to existing $z \in \mathcal{Z}$.

13

For TVaR$_\alpha$, all scenarios in the left tail need to be considered so we modify the criterion in Equation (22) to instead use weights (keeping everything else the same, including the use of $\varepsilon = s(\hat{R}_k^{HD})$)

$$W_k^{\text{TVaR}}(z; \hat{R}_k^{HD}) \doteq \frac{1}{\sqrt{2\pi(s_k^2(z) + s^2(\hat{R}_k^{HD}))}} \Phi\left(\frac{\hat{R}_k^{HD} - m_k(z)}{\sqrt{s_k^2(z) + s^2(\hat{R}_k^{HD})}}\right), \tag{23}$$

where $\Phi(\cdot)$ denotes the standard Gaussian cdf. The weights W^{TVaR} increase as z becomes deeper in the tail (i.e. $\hat{R}_k^{HD} - m_k(z)$ is more positive), while still compensating for uncertainty.

Note that a further option is to consider a convex combination $W_k = \alpha W_k^{\text{VaR}} + (1 - \alpha)W_k^{\text{TVaR}}$, where $0 < \alpha < 1$, combining the objectives of correctly identifying the cutoff for tail scenarios, with accurately predicting all tail losses. In particular, auxiliary experiments suggest that this choice with $\alpha \in [0.1, 0.3]$ improves ST-GP for learning TVaR. For brevity, we only present the results for $\alpha = 1$ (VaR) and $\alpha = 0$ (TVaR).

4.2 Expected Improvement Criterion

An alternate route based on Bect et al. [4] is to analyze the random variables $\mathbb{1}_{\{\hat{f}_k(z^n) \leq L\}}$ which have variance $\mathbb{Var}(\mathbb{1}_{\{\hat{f}_k(z^n) \leq L\}}|\mathcal{D}_k) = p_k(z^n)(1 - p_k(z^n))$, where

$$p_k(z^n) \doteq \mathbb{P}(\hat{f}_k(z^n) \leq L|\mathcal{D}_k) = \Phi\big((L - m_k(z^n))/s_k(z^n)\big).$$

These variances are low when $p_k(z^n)$ is close to 0 or 1, i.e. when $\hat{f}_k$ has strong understanding of which side $\hat{f}_k(z)$ is with regard to L. This motivates to minimize the expected level set uncertainty across all $\mathcal{Z}$, $\sum_n p_{k+1}(z^n)(1 - p_{k+1}(z^n))$, conditional on adding inner simulations at a chosen z^m. As in the previous section, we substitute $\hat{R}_k^{HD}$ for L and consider the look-ahead version $P_k(z^n; z^m)$ of $p_{k+1}(z^n)$ assuming replications are added to scenario z^m, cf. (21),

$$P_k(z^n; z^m) = \Phi\left(\frac{\hat{R}_k^{HD} - m_k(z^n)}{V_k(z^n; z^m)}\right). \tag{24}$$

The resulting Expected Contour Improvement criterion to minimize is

$$\widehat{\mathcal{H}}_k^{\text{VaR}, ECI}(z) \doteq \frac{1}{N}\sum_{n=1}^{N} P_k(z^n; z)(1 - P_k(z^n; z)). \tag{25}$$

In the sequel this allocation rule in combination with a GP emulator is labelled SE-GP.

For TVaR, the analogue of (25) would be to minimize the predictive uncertainty of the level set $\{z : \hat{f}_{k+1} \leq L\}$. From [26], this can be achieved through the acquisition function

$$\widehat{\mathcal{H}}_k^{\text{TVaR}, ECI}(z) \doteq \frac{1}{N}\sum_{n=1}^{N} P_k(z^n; z). \tag{26}$$

However, note that $\widehat{\mathcal{H}}^{\text{TVaR}, ECI}$ in fact tends to neglect scenarios satisfying $m_k(z) \ll \hat{R}_k^{HD}$ because in such cases $P_k(z^n; z) \approx p_k(z) \approx 1$ so minimal reduction in uncertainty is achieved by sampling the extreme loss scenarios. Thus, (26) ends up behaving similar to (25) and does not explore enough of the left tail as needed for proper TVaR estimation.

14

4.3 Dynamic Allocation Designs

Adding inner simulations in batches allows the possibility of sampling in parallel several different outer scenarios. This can be advantageous in anticipating the *global* updating effect of running new inner simulations. Such parallel updating also offers a bridge between the sequential approaches and the fixed-stage methods, such as the 3-stage approach of [27].

Let Δr_k be the budget for stage k. We wish to choose $\{r_k'^n\}$ to minimize the posterior estimator variance $s^2(\hat{R}_{k+1})$ in Equation (18) subject to the constraints $\sum_n r_k'^n = \Delta r_k$, and $r_k'^n \geq 0$. To do so, we extend the solution of Liu and Staum [27], who considered the specific case when r_k^n is constant in both k and n, to the sequential design context where r_k^n varies among rounds k and locations z^n. Conditioning on a set of $r_k'^n$'s and writing out the look-ahead variance,

$$s_{k+1}^2(\hat{R}_{k+1}) = \hat{\mathbf{w}}_{k+1}(\mathbf{C} - \mathbf{C}(\mathbf{C} + \boldsymbol{\Delta}_{k+1}^{cand})^{-1}\mathbf{C})\hat{\mathbf{w}}_{k+1}^T \tag{27}$$

where $\hat{\mathbf{w}}_{k+1}$ are the estimates of the weights in (5). Note that $s_{k+1}^2(\hat{R}_{k+1})$ is driven by each r_{k+1}^n through the look-ahead noise matrix $\boldsymbol{\Delta}_{k+1}^{cand}$ that has diagonal entries $\frac{\tau^2(z^n)}{r_k^n + r_k'^n}$. To proceed, we freeze the weights $\hat{w}_{k+1}^n = \hat{w}_k^n$, whereby the key calculation involves expressing the matrix inverse $(\mathbf{C} + \boldsymbol{\Delta}_{k+1}^{cand})^{-1}$ in terms of r'^n's. This is done in Lemma 1 in Appendix A. The respective proof further shows that minimizing (27) is then equivalent to minimizing

$$\mathbf{u}_k \boldsymbol{\Delta}_{k+1}^{cand} \mathbf{u}_k^T \qquad \text{where} \qquad \mathbf{u}_k^T \doteq (\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{C}\hat{\mathbf{w}}_k^T. \tag{28}$$

Solving this minimization problem can be done using a *pegging algorithm*, see e.g. [7]. As before, for the $\tau^2(z^n)$ terms in $\boldsymbol{\Delta}_k$ and $\boldsymbol{\Delta}_{k+1}^{cand}$ we use the `hetGP` estimates.

Note that (28) is predicated on *adding* replicates to scenarios with existing simulations since otherwise $\boldsymbol{\Delta}_{k+1}^{cand}$ would be undefined for $r_k'^n = 0$. Thus, to implement (28) we first create the allocation subset $\mathcal{Z}_k^{SV}$ of scenarios to optimize over so that in (28) the index n runs through $\{n : z^n \in \mathcal{Z}_k^{SV}\}$. We found that constructing $\mathcal{Z}^{SV}$ based on the ST-GP weights works well (see Section 5.3 below which uses the same construction for screening scenarios for ST-GP and SE-GP). If necessary, prior to minimizing (28) we then add single replicates to any scenarios in $\mathcal{Z}_k^{SV}$ that have $r_k^n = 0$.

The above algorithm, which we label SV-GP (for Variance minimization) in what follows, based on solving (28) makes three approximations. First, it modifies the minimization problem by using a matrix approximation in Lemma 1. Second, it treats noise variances as known (specifically it uses same estimate of $\tau^2(z)$ across stage k and $k+1$). Third, it treats the GP emulator as fixed, i.e. it does not take into account that the hyperparameters are themselves evolving in k. Due to all of the above, the resulting allocation $r_k'^n$ is sub-optimal. This means that while the method is providing a recipe to allocate arbitrary simulation budget, it cannot entirely replace the sequential rounds. Indeed, this is the basic shortcoming of a fixed-stage approach like that of [27]: by taking only a few stages, extra onus is placed on the initialization and proper fine-tuning of the GP surrogate. As we show below, the ultimate performance of SV-GP is not materially better than the more "naive" one-scenario-at-a-time strategies of the previous section. Since the approximations in (28) diminish as each r_k^n, $n = 1, \ldots, N$ increases, SV-GP tends to perform best at the later stages. It also implies that it may be beneficial to increase batch amounts Δr_k as k grows.
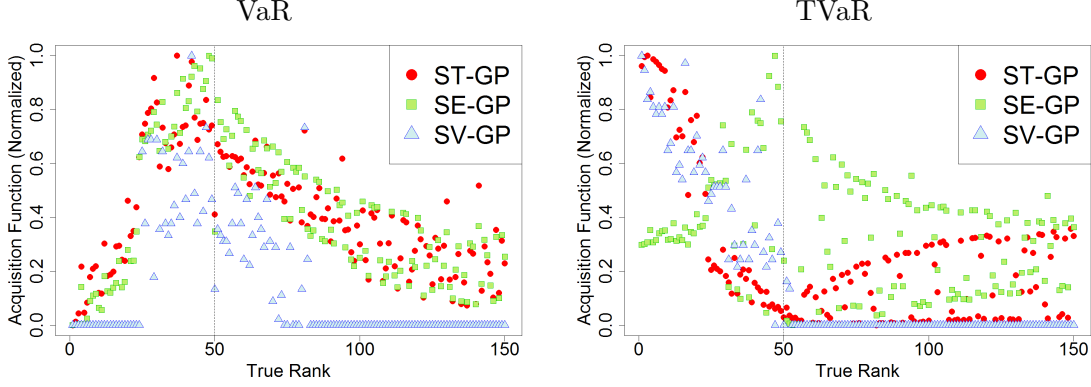
Figure 1: Normalized acquisition functions for ST-GP, SE-GP and SV-GP on the same simulation output, namely after round $k = 20$, using SR-GP-2 and the 2-D Black-Scholes case study. Here, "acquisition function" for SV-GP refers to the allocation results from the minimization problem in Equation (28). For visualization, ST-GP and SE-GP criteria are changed in sign so that they allocate Δr_k to the point with the *highest* value; all $\mathcal{H}$'s are normalized to be in the range $[0, 1]$. The vertical dashed lines indicate the true quantile scenario $\mathfrak{q}_\alpha$.

To illustrate the various acquisition functions, Figure 1 shows normalized values of $\widehat{\mathcal{H}}_k(z^n)$ for ST-GP, SE-GP and SV-GP. For the latter, we take $\mathcal{H}_k^{SV}(z^n) \equiv r'^n$, i.e. the next-stage allocation amounts. The underlying model comes from the case study in Section 6 below. We observe that for VaR, all $\widehat{\mathcal{H}}$ target the quantile $\mathfrak{q}_\alpha$, with SV-GP more strongly focused in this region. Recall that ST-GP and SE-GP will pick z^{k+1} as the maximizer of the shown acquisition functions, while SV-GP divides Δr_k according to the indicated $r_k'^n$. For TVaR, ST-GP and SV-GP target the deep tail (scenarios with worst portfolio losses), while SE-GP still focuses on the quantile region.

4.4 Rank-Based Allocation

A simpler approach is to allocate inner simulations based on the posterior means $m_k(z^n)$. Specifically, for given parameters L and U satisfying $L \le \alpha N \le U$, the SR-GP ("Sequential Rank") algorithm allocates uniformly to all scenarios with rank between L and U: $\mathfrak{R}_k^{SR-GP}(L, U) \doteq \{z^n \in \mathcal{Z} : m_k(z^n) \in [m_k^{(L)}, m_k^{(U)}]\}$. Thus, $r_k'^n = \Delta r_k / (U - L)$ for $z^n \in \mathfrak{R}_k^{SR-GP}(L, U)$ and zero otherwise.

If the values of L and U are chosen conservatively, SR-GP "blankets" all scenarios in the predicted neighborhood of interest. This ensures exploration of the potential tail, but the adaptive targeting might not be localized enough. Conversely, taking $L = U = \alpha N$ yields a very aggressive SR-GP scheme for estimating VaR$_\alpha$: it greedily adds all Δr_k scenarios to the empirical quantile, i.e. scenario $\widehat{\mathfrak{q}_{\alpha k}}$. Employing this extreme as a comparator addresses the value of exploration during the sequential design. In general, SR-GP should be suboptimal compared to ST-GP, SE-GP and SV-GP since it does not take uncertainty of $\hat{f}_k$ into account. It also suffers from the a priori difficulty to reasonably set L and U. To obtain well-performing estimators in the case studies, our choices for L and U came through several iterations of tweaking to see what worked well, all depending on the case study and risk measure.
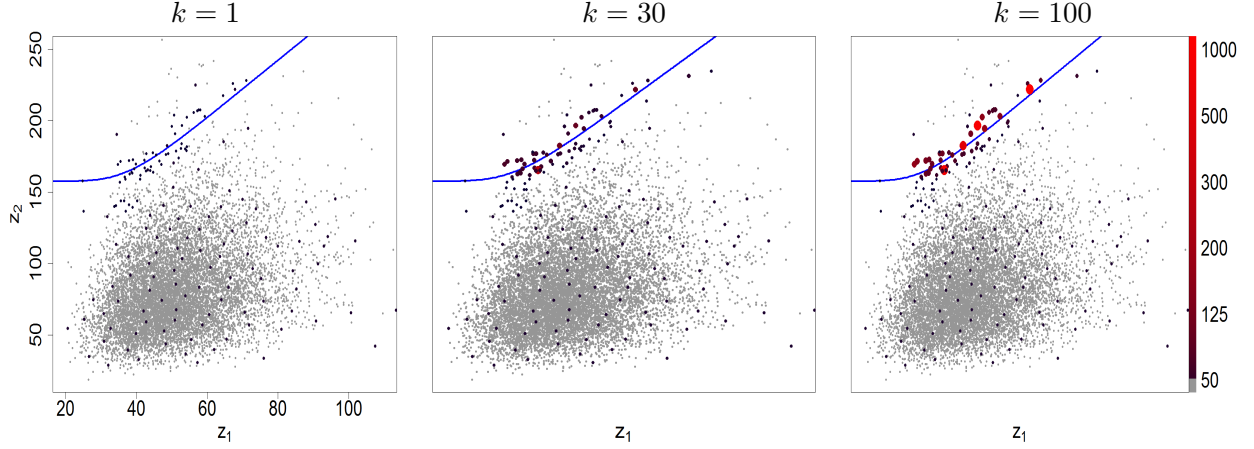
16

5 Algorithm

Figure 2: Sequential budget allocation by SV-GP at stages $k = 1, 30, 100$ to learn $\text{VaR}_{0.005}$ in the 2-D Black-Scholes case study of Section 6. The blue line indicates the true quantile contour $f^{(50)}$. Each dot represents an outer scenario z^n; the respective size and color are scaled non-linearly in r_k^n. Some scenarios receive as many as $r_k^n \approx 1200$ scenarios (total budget of $\mathcal{N} = 10^4$).

To illustrate our sequential schemes, Figure 2 visualizes the evolution of scenario allocations r_k^n over one run of the SV-GP method for VaR estimation. As k increases, the vast majority of the budget goes to the tail scenarios, highlighting the adaptive inner simulations. Thus, in the later stages very few additional scenarios are augmented to $\mathcal{D}_k$; rather the values of r_k^n are increasing for locations near the quantile $\mathfrak{q}_\alpha$. The learning of f can be observed by comparing the $k = 1$ panel where the algorithm investigated scenarios with $z_2 > 225$, and the panels with $k = 30$ and $k = 100$ where the corresponding values of r_k^n remain low after the emulator realizes that $|m_k(z^n) - R|$ is in fact large in that region. The final plot for $k = 100$ also shows how the GP's spatial covariances are taken into account: scenarios are sampled in "clusters"—centered around a single scenario with a very high r_k^n, with nearby locations benefiting from this information.

5.1 Updating the Emulator

At each stage of the algorithm, the GP emulator is updated from $\hat{f}_k$ to $\hat{f}_{k+1}$. After recording the number of replications $(r_k'^n)_{n=1}^N$ to add to each scenario according to the procedures discussed in Section 4 and generating the new data $\{y^{n,i} : i = r_k^n + 1, \ldots r_k^n + r_k'^n\}$, we have the recursive update

$$r_{k+1}^n = r_k^n + r_k'^n, \tag{29}$$

which is used for $\boldsymbol{\Delta}_{k+1}$ in (13). Next, directly re-computing m_{k+1}, s_{k+1}^2 requires inverting the $|\mathcal{D}_k| \times |\mathcal{D}_k|$ GP covariance matrix $\mathbf{C}$ which is computationally expensive. Instead, we *update* the GP model by re-using the expressions on the RHS of (13) from stage k. In our context updates come in two flavors, both of them internally incorporated in `hetGP`. First, if a new outer scenario z^{k+1} (that previously had $r_k^n = 0$) is added to $\mathcal{D}_k$ then the GP updating equations [15] can be applied,

17

utilizing the Woodbury formula for finding matrix inverse when augmenting by 1 row. Second, if only new replications are added, i.e. $|\mathcal{D}_{k+1}| = |\mathcal{D}_k|$ then we adjust $\boldsymbol{\Delta}_{k+1}$ without touching $\mathbf{C}$ [5]. Intuitively, it is sufficient to just keep track of $\bar{y}^n$, $\hat{\tau}^2(z^n)$ and r_k^n's, and the latter 3 vectors all satisfy simple updating arithmetic [10, 24]

$$\bar{y}_{k+1}^n = \frac{r_k^n \bar{y}_k^n + r_k'^n \bar{y}_k'^n}{r_k^n + r_k'^n}, \tag{30}$$

$$\hat{\tau}_{k+1}^2(z^n) = \frac{1}{(r_k^n + r_k'^n - 1)} \left((r_k^n - 1)\hat{\tau}^2(z^n) + (r_k'^n - 1)\hat{\tau}_k'^2(z^n) + \frac{r_k'^n r_k^n}{r_k^n + r_k'^n} \left(\bar{y}_k^n - \bar{y}_k'^n \right)^2 \right), \tag{31}$$

where $\bar{y}_k'^n$ is the average of the new $y^{n,i}$'s and $\hat{\tau}_k'^2$ is the corresponding sample variance. Note that updating freezes the GP hyperparameters, see Section 5.3 below.

5.2 Initialization of $\hat{f}$

Initially, we have $r^n = 0$ for all n, i.e. no data to inform us where the tail region may lie. To initialize, the typical solution is to use a small number N_{init} of *pilot* scenarios that are representative of the entire domain $\mathcal{Z}$, determined by some space-filling algorithm (e.g. Latin Hypercube sampling, LHS). Chauvigny et al. [11] provides a more elaborate method using statistical *depth functions* to identify pilot scenarios based on the geometry of $\mathcal{Z}$.

A small challenge is that LHS and other space-filling approaches are not directly applicable to a discrete scenario set which is non-uniform in space (see Figure 3). Instead we develop a minimax-style initialization procedure based on Euclidean distance $\|z - z'\|_2$ and described in Appendix B. Note that the resulting design size $N_{init} = |\mathcal{D}_0|$ is not fully pre-determined. After determining $\mathcal{D}_0$, we allocate an equal number of inner simulations to each scenario, i.e. $r_0'^n = \Delta r_0 / N_{init}$ for all $\{n : z^n \in \mathcal{D}_0\}$ to build $\hat{f}_0$. In our case studies below we take $\Delta r_0 = 0.1\mathcal{N}$ and $N_{init} = 0.01N$ so that we spend 10% of the total budget on the pilots which cover 1% of the scenarios. The $k = 1$ panel of Figure 2 illustrates this space-filling procedure by plotting the pilot scenarios (heavy black dots). It also shows the first round allocation, where locations having $r_1^n > 0$ are starting to line up along the initial contour estimate.

5.3 Implementation Details

The two most popular kernels for GP emulation are the Matérn$-5/2$ family in Equation (14) and the Gaussian family,

$$g^{Gsn}(h, \theta) \doteq \exp\left(\frac{-h^2}{2\theta^2} \right). \tag{32}$$

Note that the choice of the kernel family modifies the meaning of the hyperparameters, so direct comparison is fraught. In Section 6.4 we present numerical evidence that the choice of covariance kernel generally has only a minor impact on the resulting emulators $\hat{f}_k$.

The classical method for inferring the hyperparameters $\boldsymbol{\theta}$ and process variance σ^2 is by optimizing the marginal likelihood, either through MLE or penalized MLE, using the likelihood function based on the distributions described in Section 3.1. Either case leads to a nonlinear optimization problem. In a sequential setting, the hyperparameters are ideally refitted at each stage to improve

the Likelihood function as more data comes in. However, this is computationally impractical since evaluation of the likelihood requires $\mathcal{O}(|\mathcal{D}_k|^3)$. Alternatives to this are to refit the hyperparameters at certain points in the algorithm, e.g. after 10%, 20%, ..., 90% (or with a nonlinear schedule, such as 2%, 4%, 8%, ...) of the budget has been depleted. The latter scheme has the advantage of refitting more frequently earlier when there is less certainty about the hyperparameters, and when the refits are cheaper.

The above logic also implies preference for algorithms that keep $|\mathcal{D}_k|$ small. In that sense a more targeted algorithm like ST-GP is preferred over SV-GP that by its nature spreads inner simulations across many scenarios. A related overhead concerns predicting $\hat{f}_k$ on $\mathcal{Z}$, i.e. evaluating $m_k(z^{1:N})$ and $s_k(z^{1:N})$, which is required by all the acquisition functions. This carries $\mathcal{O}(|\mathcal{D}_k|^2 \cdot |\mathcal{Z}|)$ effort, once again showing the cost of a large design $\mathcal{D}_k$. It also leads to the trick of reducing $\mathcal{Z}$ to a candidate set $\mathcal{Z}_k^{cand}$ by screening scenarios that are far from the tail. One way to screen is to re-use the weights $W_k(z^n)$ from Equations (19) and (23) to assess relevance of scenarios. Specifically, in our case studies below we determine the candidate set $\mathcal{Z}_k^{cand}$ at stage k via

$$\mathcal{Z}_k^{cand} \doteq \{z^m \in \mathcal{Z} : W_k^{\mathrm{VaR}}(z^m) / (\sum_{n=1}^{N} W_k^{\mathrm{VaR}}(z^n)) > 10^{-3}\},$$

for VaR and $\mathcal{Z}_k^{cand} \doteq \{z^n \in \mathcal{Z} : W_k^{\mathrm{TVaR}}(z^n) > 10^{-3}\}$ for TVaR. Thus only above outer scenarios are considered when evaluating the acquisition functions (methods ST-GP, SE-GP, SV-GP). As an illustration, on a typical run of the first case study, this screening shrunk the prediction set from $N = 10000$ scenarios to $|\mathcal{Z}_1^{cand}| = 528$ (657 for TVaR) after the first round, reducing the overhead of predicting the GP surrogate on $\mathcal{Z}_1^{cand}$ by a factor of ≈ 20. Furthermore, the candidate sets $|\mathcal{Z}_k^{cand}|$ shrink as k grows, as the GP better learns the tail scenarios.

In the examples below we used a constant stage-wise budget $\Delta r_k \equiv \Delta r$; an embellished version could easily include changing batch sizes over rounds, for example smaller batches in earlier rounds where the GP has high global uncertainty. Choosing batch size also determines the total number of rounds K, influencing GP regression and prediction overhead, as well as the computation cost of SUR acquisition functions. In general we find (cf. Section 6.1) that the algorithms focus on only a small number of scenarios which means that the same z^n's get repeatedly picked by our sequential criteria. As a result, a large batch size, say $\Delta r = 0.1\mathcal{N}_1$, offers little performance loss compared to a smaller $\Delta r = 0.01\mathcal{N}_1$, and reduces the overhead of selecting z^{k+1}.

Returning to the GP emulator itself, we found that a simple trend function $\mu(\cdot)$ in (12) is sufficient for de-trending. While the choice of $\mu(\cdot)$ has little impact on the prediction $\hat{f}_k$, it helps the nonlinear optimization routine in fitting the hyperparameters, and is also beneficial in the very beginning of the sequential stages. Note that the trend modifies somewhat the spatial dependence of the response and hence the θ_j hyperparameters. A parametric mean function can also be specified via basis functions with respective coefficients to be fitted, so that $\mu(z) = \beta_0 + \sum_j \beta_j h_j(z)$. In this case, the β's are estimated simultaneously with the other hyperparameters, an approach known as Universal Kriging [33]. In our experience, in most financial settings $\mu(\cdot)$ can be either easily identified as the intrinsic value of the portfolio (like in our first case study), or taken to be a constant β_0 (fitted as part of the GP, see the second case study).

5.4 Comparison with other Approaches

To benchmark the proposed algorithms defined in Section 4, we compare to several alternatives. We primarily focus on other regression-based approaches and concentrate on quantifying the role of (i) adaptive budget allocation; and (ii) sequential approaches as compared to simpler 1-, 2- or 3-stage methods. A summary of these benchmarks, as well as the procedures discussed previously in the section are given in Table 1.

1. LB: a perfect information method used as a Lower Bound. We assume that the tail/quantile scenarios are known a priori, and look to minimize MSE of $\hat{R}$. For VaR this corresponds to minimizing the posterior variance at $\mathfrak{q}_\alpha$ which is trivially achieved by allocating the entire budget $\mathcal{N}$ to the true quantile scenario. In that case there is no GP surrogate and the estimator variance is the Monte Carlo averaging error, i.e. $\tau^2(\mathfrak{q}_\alpha)/\mathcal{N}$. For TVaR$_\alpha$, we allocate the budget $\mathcal{N}$ uniformly among the true tail $\{z^n : f(z^n) \leq f^{(\alpha N)}\}$ and fit a GP to the results.

2. A3-GP: The Adaptive 3-stage algorithm of Liu and Staum [27]. The first stage simulates from space-filling pilot scenarios similar to our approach. The second stage allocates uniformly across a screened candidate set $\mathcal{Z}_1^{cand}$. The third stage then solves for r'^n to minimize variance of $\hat{R}_2$ like in SV-GP and Lemma 1. We follow their suggestions, with $0.02N$ stage-2 design locations, and stage-3 budget of $\mathcal{N}_2 = 0.7\mathcal{N}$.

3. U2-GP: A 2-stage approach. After a space-filling first stage as in Section 5.2, the remaining budget is allocated uniformly among the lowest $2\alpha N$ scenarios. This is the simplest version of an adaptive allocation with $r_1'^n = \mathcal{N}_1/(2\alpha N)\mathbb{1}_{\{m_0(z^n) \leq m_0^{(2\alpha N)}\}}$. Note that U2-GP can be seen as a version of SR-GP with $K = 2$ rounds and $L = 1, U = 2\alpha N$. Comparing to this approach quantifies the gain of multi-stage procedures.

4. U1-GP: A 1-stage approach which uniformly allocates $\mathcal{N}/|\mathcal{Z}|$ to each $z \in \mathcal{Z}$, fits a GP surrogate $\hat{f}_0$ to the resulting design and uses $m_0^{1:N}$ to estimate $\hat{R}$. This is the crudest comparator that has no adaptive allocation but still employs spatial smoothing. We find that fitting a GP to the entire collection of outputs is computationally unfeasible for large scenario sets, $N \gg 1000$, so instead we fit to $\{z^n : y^n \leq y^{(2000)}\}$, which yields similar overhead times to ST-GP and still offers significant improvement over no GP smoothing.

We also consider two non-GP methods to investigate the importance of spatial smoothing. The first is called U1-SA, which is the same as U1-GP but uses the sample means $\bar{y}^n$ and empirical variances $\hat{\tau}^2(z^n)$ in place of GP-based posterior means. This is the "vanilla" nested simulation method. The second is a ranking-and-selection (R&S) approach, introduced in Broadie et al. [8] for the purpose of comparing $f^{1:N}$ with a given level L. With our notation, at stage k this BR-SA algorithm allocates Δr_k simulations to the scenario z^{k+1} which minimizes $\mathcal{H}_k^{BR}(z^n) \doteq r_k^n \cdot |\bar{y}_k^n - L|/\tau(z^n)$. The above acquisition function compares the scenario sample average to L, normalizing by the respective sample uncertainty $\tau(z^n)/r_k^n$ which resembles $\mathcal{H}^{ECI}$ in (25) but without having an emulator. For our purposes we replace L with $\hat{R}_k^{HD}$. Also, we follow the suggestion of [8] to estimate local noise variance by a weighted $\tilde{\tau}(z^n)$ based on the aggregate variances gathered,

Name	Description	Parameters
LB	Monte Carlo under perfect information	TVaR allocates uniformly to true tail scenarios
ST-GP	Targeted MSE criterion (22)	$\varepsilon_k = s(\hat{R}_k^{HD})$
SE-GP	Expected Level Improvement criterion (25)	
SV-GP	Batch Variance minimization (Section 4.3)	
SR-GP-1	Uniform on $\{z^n : m_k^{(L)} \le m(z^n) \le m_k^{(U)}\}$	L=50, U=50 (VaR), L=1,U=50 (TVaR)
SR-GP-2		L=26, U=75 (VaR), L=1, U=75 (TVaR)
A3-GP	3-stage from [27]	70% of budget for stage 3; $r_2^n \equiv 10$
U2-GP	Two-stage	SR-GP with $L = 1, U = 100$ and $K = 2$
U1-GP	Uniform allocation	GP fit to the bottom 20% of z^n's based on $\bar{y}_1^{1:N}$
U1-SA	Uniform allocation	Uses sample averages $\bar{y}^{1:N}$ as output
BR-SA	R&S algorithm from [8]	

Table 1: Comparator approaches as described in Section 5.4. All approaches (except U1-GP) use the same initialization procedure described in Algorithm 1 with $N_{init} = 0.01N$ stage-1 scenarios. The budgeting parameters for the sequential methods are chosen to yield $K = 100$.

corresponding to a very simple homoskedastic smoother:

$$\tilde{\tau}_k^2(z^n) = \frac{r_k^n}{r_k^n + \tilde{r}}\hat{\tau}_k^2(z^n) + \frac{\tilde{r}}{r_k^n + \tilde{r}}\bar{\tau}_k^2,$$

where $\bar{\tau}_k^2 = \frac{1}{N'}\sum_{n=1}^{N'}\hat{\tau}_k^2(z^n)$ is the average over all empirical variances. Here, $\tilde{r}$ is the smoothing parameter, and we take $\tilde{r} = 5$ following [8]. A comparison with these non-GP benchmarks is discussed in Section 6.3.

Remark. Note that following above "recipe" one may combine any of the introduced acquisition functions with a sample-averages based approach, yielding, say, ST-SA or SV-SA approaches. As explained, a limitation of not having an emulator is inability to properly forecast $\tau^2(z)$ at scenarios that do not have enough inner simulations. From the other direction, we recall the Least Squares Monte Carlo methods (like in Bauer et al. [3]) which use spatial smoothing but a non-sequential allocation. We do not compare to the latter since our focus is on experimental design rather than purely the regression/smoothing step.

6   Case Study: Black Scholes Option Portfolio

We begin with a two–dimensional example where $f(z)$ can be computed exactly. Working in 2-D with a known f allows for easy visualization of the algorithms, and provides exact error calculations. Consider a portfolio whose value is driven by two risky assets $Z_t \equiv (S_t^1, S_t^2)$ that have Geometric Brownian motion dynamics:

$$dS_t^1 = \beta S_t^1\, dt + \sigma_1 S_t^1\, dW_t^{(1)},$$
$$dS_t^2 = \beta S_t^2\, dt + \sigma_2 S_t^2\, dW_t^{(2)}.$$

Above the W^i are correlated Brownian motions under the risk neutral measure $\mathbb{Q}$ with $d\langle W^{(1)}, W^{(2)}\rangle_t = \rho\, dt$ and β is the constant interest rate. Our model parameters are summarized in Table 2. The portfolio consists of Call options: long 100 $K_1 = 40-$strike Calls on S^1 and short 50 $K_2 = 85-$strike

Calls on S^2. We work with a risk horizon $T = 1$ year, and target VaR_α and TVaR_α risk measures with $\alpha = 0.005$. By risk-neutral pricing, the value of the portfolio at T and starting with $z \equiv (z_1, z_2) \in \mathbb{R}_+^2$ is

$$f(z) \doteq \mathbb{E}^{\mathbb{Q}} \left[100 e^{-\beta(T_1 - T)} \left(S_{T_1}^1 - 40 \right)_+ - 50 e^{-\beta(T_2 - T)} \left(S_{T_2}^2 - 85 \right)_+ \Big| (S_T^1, S_T^2) = z \right]. \tag{33}$$

Asset	Position	Initial Price S_0^i	Strike K_i	Maturity T_i	Volatility σ_i
S^1	100	50	40	2	25%
S^2	-50	80	85	3	35%
	Correlation $\rho = 0.3$			Interest Rate $\beta = 0.04$	

Table 2: Parameters of the 2-D Case Study for a Black-Scholes portfolio on stocks S^1 and S^2.

A contour plot of f, obtained from the Black Scholes formula, is shown in Figure 3. Observe that $f(z)$ is most negative in the upper-left corner, while $\tau^2(z)$ is increasing in both z_1 and z_2, so that additional focus may be spent toward the upper-right corner where uncertainty is largest and the scenarios are more scarce. The figure also exhibits the scenario set $\mathcal{Z}$ which we generated as a *fixed* sample from the bivariate log-normal distribution of (S_T^1, S_T^2). Note that here $\mathcal{Z}$ was generated using $\mathbb{Q}$-distribution, so the dynamics of the factors on $[0, T]$ and $[T, T_i]$ are the same, i.e. the physical and risk-neutral measures coincide. This is solely for a simpler presentation of the case-study; in our experience the role of $\mathcal{Z}$ is secondary to the other considerations. The red line in Figure 3 shows the true (relative to the shown $\mathcal{Z}$) quantile loss $f^{(\alpha N)} = f(\mathfrak{q}_\alpha) = -4052.02$, indicating the region that the methods are supposed to target.
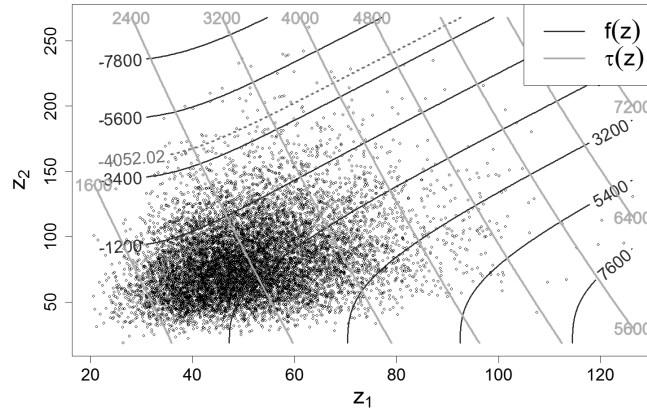

Figure 3: The true portfolio value $f(z)$ for the 2-D Black-Scholes option case study, along with the respective simulation standard deviation $\tau(z)$. In red is the true quantile loss $f^{(50)} = -4052.02$. The point cloud represents the scenario set $\mathcal{Z} = \{z^n, n = 1, \ldots, 10000\}$.

In this case study, the outputs $y^{n,i}$ are obtained by simulating the log-normal values of S_2^1, S_3^2 conditional on $(S_1^1, S_1^2) = (z_1, z_2)$ and plugging them into the payoff

$$Y = 100 e^{-\beta(T_1 - T)} \left(S_{T_1}^1 - 40 \right)_+ - 50 e^{-\beta(T_2 - T)} \left(S_{T_2}^2 - 85 \right)_+, \tag{34}$$

where $T = 1$, $T_1 = 2$ and $T_2 = 3$. We proceed to compare all the methods listed in Table 1 using the global experiment parameters of $N = 10^4$, $\mathcal{N} = 10^4$, so that $\alpha N = 50$. Note that since $\mathcal{N} = N$, U1-GP allocates a single inner simulation per scenario. The fully sequential schemes use $N_{init} = 0.01N = 100$ with $r_0^n = 10$ during initialization (i.e. $\mathcal{N}_1 = \mathcal{N} - r_0 N_{init} = 9000$), and $K = 100$ stages, so that $\Delta r_k = \mathcal{N}_1/K = 90$ during all other rounds. The GP hyperparameters are refitted at stage $k = 10, 20, \ldots, 100$ for each sequential method, and after each stage for A3-GP and U2-GP.

Since we have the exact value-at-risk $R^{\text{VaR}} \doteq f^{(50)}$ and Tail VaR $R^{\text{TVaR}} \doteq \frac{1}{50} \sum_{n=1}^{50} f^{(n)}$, the bias and squared error (SE) of a given final estimate $\hat{R}_K$ for either VaR or TVaR are simply

$$\text{bias}(\hat{R}_K) \doteq \hat{R}_K - R, \qquad \text{and} \qquad \text{SE}(\hat{R}_K) \doteq \left(\hat{R}_K - R \right)^2.$$

For the mean function $\mu(\cdot)$ of the GP in (12) we took the intrinsic value of the portfolio at T, that is,

$$\mu(z_1, z_2) = 100e^{-0.04}(z_1 - 40)_+ - 50e^{-2 \cdot 0.04}(z_2 - 85)_+. \tag{35}$$

The above choice was compared to a simpler constant mean function $\mu(z) = \beta_0$ with β_0 fitted along with other GP hyperparameters; the latter yielded no statistically significant differences in the resulting performance of the schemes although it did increase the GP uncertainties $s_k(z^n)$.

6.1 Comparing Algorithms

Assessment of the algorithms is done through performing 100 macro-replications $m = 1, \ldots, 100$ (i.e. repeating the whole procedure with a fresh set of sampled $y^{n,i}$'s) for both VaR and TVaR with the fixed set $\mathcal{Z}$, obtaining estimates $\hat{R}_K^{[1]}, \ldots, \hat{R}_K^{[100]}$. This yields a true sampling distribution of the estimators from various algorithms, controlling for the intrinsic variability of inner simulations. These outputs are used to compute bias, variance, and SE, and the results are illustrated through several tables and figures. Detailed visualization of sequential methods is provided in Section 6.2.

Table 3 reports (i) $SD(\hat{R}_K^{[1:100]})$ which is the empirical standard deviation of $\hat{R}_K$ over the 100 macro replications, (ii) the average estimated GP posterior uncertainty associated to $\hat{R}_K$: $\bar{s} \doteq \frac{1}{100} \sum_{m=1}^{100} s(\hat{R}_K^{[m]})$, (iii) the root mean squared error relative to the ground truth, $RMSE \doteq \sqrt{\frac{1}{100} \sum_{m=1}^{100} (\hat{R}_K^{[m]} - R)^2}$ along with (iv) average $|\mathcal{D}_K|$, the number of distinct outer scenarios chosen by the algorithm. Note that for methods other than LB and U1-GP, $|\mathcal{D}_K| - 100$ (where $100 = N_{init}$ is the number of pilot stage-0 scenarios) is the number of locations chosen after initialization. Figures 4 and 5 show boxplots of the resulting distributions for $\hat{R}_K^{[m]}$ and $s(\hat{R}_K^{[m]})$, where the horizontal line is the true risk measure R obtained from the analytic Black-Scholes computation. For VaR we recall that our estimators are expected to converge to the Harrell-Davis estimator $R^{\text{VaR},HD}$ which is used as the ground truth in our discussion below. In the present example the difference relative to the true quantile was about 20, $f^{(\alpha N)} = -4052.02$, $R^{HD} = -4032.21$.

We begin discussion with Value-at-Risk. The uniform allocation (U1-GP) results are not plotted since they are far beyond axis limits; the tables show that its RMSE is more than 60 times larger than that of LB. All other methods offer clear improvement over traditional nested Monte Carlo. Comparing in the order of U1-GP, U2-GP and A3-GP roughly shows the improvement thanks to

	VaR$_{0.005}$				TVaR$_{0.005}$			
	$SD(\hat{R}_K^{HD})$	$\bar{s}$	RMSE	$|\mathcal{D}_K|$	$SD(\hat{R}_K)$	$\bar{s}$	RMSE	$|\mathcal{D}_K|$
LB	44.35	40.42	46.77	1	47.29	53.48	47.42	1
ST-GP	50.57	48.55	50.59	121.52	59.12	55.17	61.46	118.27
SE-GP	50.48	50.71	74.03	116.12	93.36	87.83	95.70	111.79
SV-GP	56.50	48.28	60.53	305.03	55.78	54.76	56.65	163.08
SR-GP-1	63.27	61.90	69.74	112.43	61.48	55.34	61.86	165.27
SR-GP-2	50.45	49.82	50.52	180.97	61.66	61.56	62.13	193.46
A3-GP	61.07	54.18	60.83	292.83	63.57	59.92	63.18	297.44
U2-GP	68.76	55.91	68.47	194.55	64.92	67.77	64.87	194.64
U1-GP	695.33	560.52	2965.05	10^4	909.17	700.43	3003.07	10^4

Table 3: For the 2-D Black Scholes case study, sample standard deviation (SD) over 100 macro-replications, average GP posterior standard deviation $\bar{s}$, and RMSE for each approach for $\hat{R}_K$, as well as average final design sizes. Description of the methods is in Table 1.
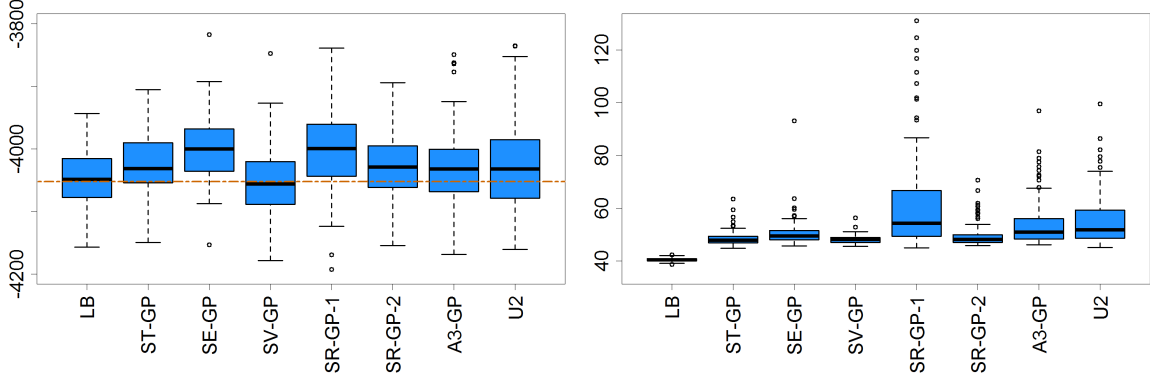

Figure 4: VaR estimation for the 2-D Black Scholes case study. Left boxplots display the distribution of the final $\hat{R}_K^{\mathrm{VaR}}$ estimates; on the right is distribution of the corresponding GP standard deviation $s(\hat{R}_K^{\mathrm{VaR}})$. Results are based on 100 macro-replications for each approach.

increasing the number of stages, with the latter gaining about 20% in RMSE reduction relative to U2-GP. Going to a sequential scheme with 100 stages we gain another 15-20% improvement – compare ST-GP to A3-GP.

A few other comparisons can be made. Between the two SUR-type strategies, SE-GP does not perform well relative to ST-GP, suggesting that the latter acquisition function is more appropriate for quantile learning, likely due to its ability to account for uncertainty in the estimated $\hat{R}_K^{HD}$. For the rank based methods, SR-GP-2 doing well (competitive with ST-GP) shows that the choices of $L = 26, U = 75$ work well for VaR$_{0.005}$ in this case study, while the poor performance of SR-GP-1 is indicative of its narrow focus on the empirical quantile. By being overly aggressive, SR-GP-1 may completely miss some scenarios that constitute R^{HD}, which contributes to more bias and dispersion across runs.

Overall, in terms of RMSE, the best performing are ST-GP, SV-GP and SR-GP-2. From the
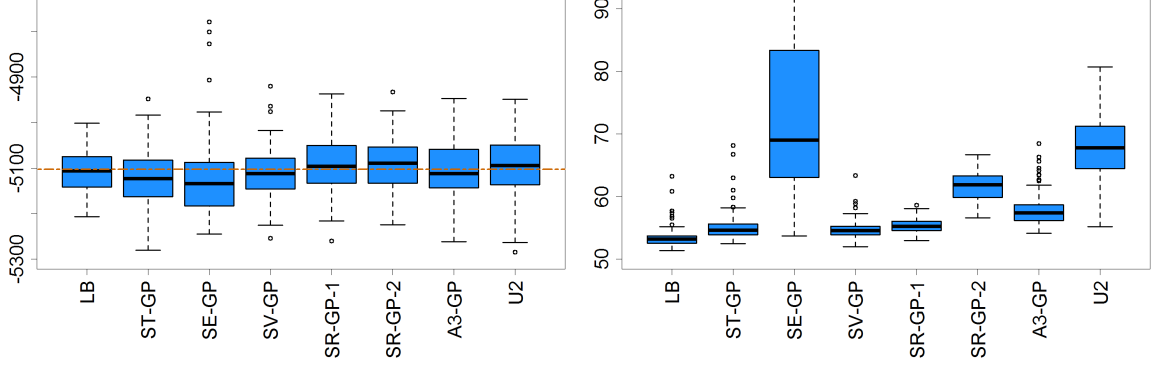
Figure 5: TVaR estimation for the 2-D Black Scholes case study. Left boxplots display the distribution of the final $\hat{R}_K^{\mathrm{TVaR}}$ estimates; on the right is corresponding GP standard deviation $s(\hat{R}_K^{\mathrm{TVaR}})$. Results are based on 100 macro-replications for each approach.

boxplots we see that all methods (except LB which as expected is unbiased) are biased high, i.e. under-report capital requirements. Of note, SE-GP and SR-GP-1 have relatively high biases and also larger $SD(\hat{R}_K)$. Note that without spatial smoothing, quantile estimates are biased *low* since empirical quantiles are more extreme relative to the ground truth. Through spatial averaging, the emulator pulls the bias the other way.

Table 3 also compares $s(\hat{R}_K^{[m]})$ against $SD(\hat{R}_K)$. The former is the internal emulator-based estimate of the standard error for $\hat{R}$, while the latter is the observed sampling standard deviation across the macro-replications. There is no way to calculate $SD(\hat{R}_K)$ a priori, and we hope that $s(\hat{R}_K)$ can act as a proxy. This ability to accurately report standard errors is an important part of the emulation. Of course, $s(\hat{R}_K^{[m]})$ is itself random, so in the Table we report its average $\bar{s}$ and Figure 4 shows its own sampling distribution. Thus, the goal is to have $s(\hat{R}_K^{[m]})$ stable across runs and with mean close to $SD(\hat{R}_K)$. We do observe that all methods have $\bar{s}$ reasonably close to $SD(\hat{R}_K^{[1:100]})$, with the biggest discrepancies occurring with SV-GP and A3-GP. Both of these methods are designed to minimize $s(\hat{R}_k)$, which may cause these values to be biased low. From the boxplots we note that SV-GP has the most stable $s(\hat{R}_K)$ across runs, while SR-GP-1, A3-GP, and U2-GP offer least reliable standard error estimates.

For Tail Value-at-Risk, the results are nearly identical. We again witness poor performance of SE-GP (RMSE 2.01 times larger than LB) which confirms that the respective acquisition function $\mathcal{H}^{ECI}$ is inadequate. Also note that in the TVaR context, the more aggressive SR-GP-1 outperforms SR-GP-2. This illustrates that SR-GP requires careful fine-tuning of the L and U values.

A final visual comparison of the sequential methods is provided through fan plots in Figure 6 which illustrates evolution of the estimated risk measure $\hat{R}_k$ as the rounds k progress. This is one of the major advantages of a sequential procedure which allows "online" use of the algorithm. Thus the user can monitor $\hat{R}_k$ for example to judge the convergence, adaptively stop the simulations, or report interim estimates. The fan plots show quantiles (in terms of m) over macro replications of $R_k^{[m]}$ as a function of k. Most methods are initially biased high, the bias vanishes as k increases and sampling variance decreases. SV-GP reduces bias the fastest. Of note, SR-GP-1 does not converge
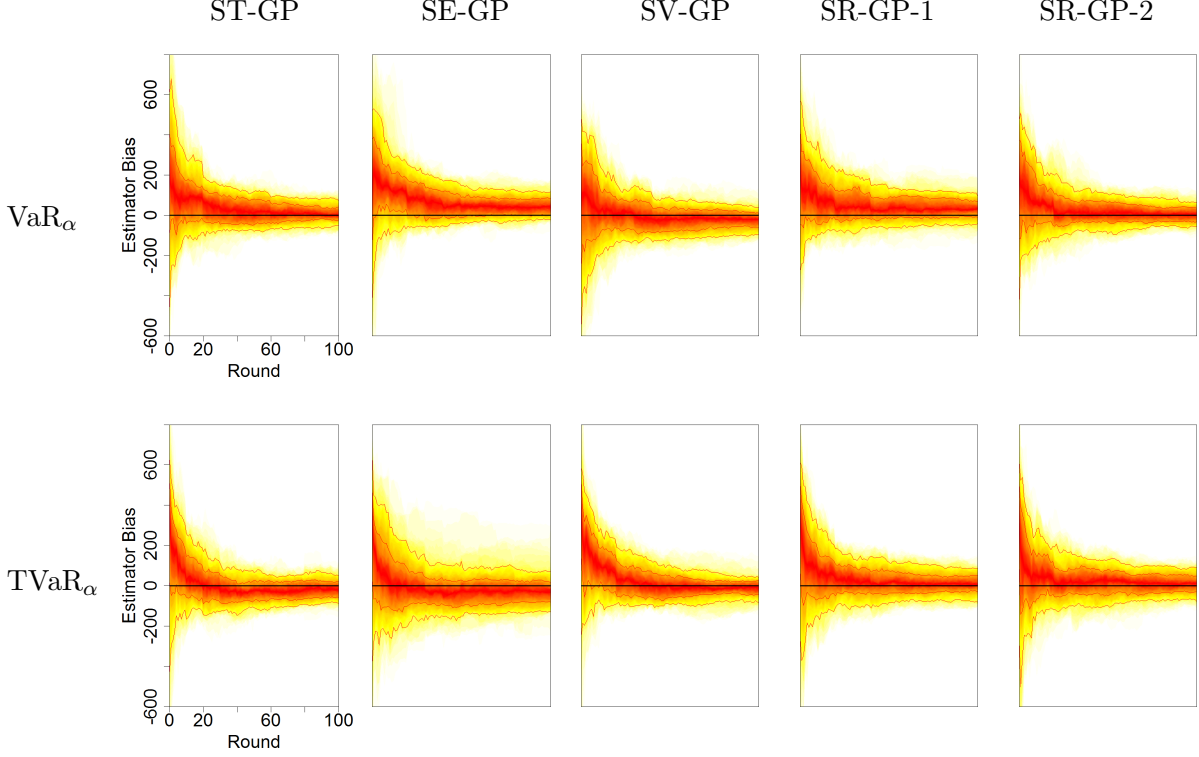
Figure 6: For the 2-D Black Scholes case study, fan plots describing evolution of $\hat{R}_k^{HD,\text{VaR}}$ (top row) and $\hat{R}_K^{\text{TVaR}}$ (bottom) as budget is spent. Red lines correspond to the 0.1, 0.25, 0.5, 0.75, and 0.9 componentwise quantiles of $\hat{R}_k^{[m]}$ over the 100 algorithm runs.

on some runs, highlighting the importance of exploration.

For TVaR we observe that $bias(\hat{R}_k)$ is generally low already after a few rounds, indicating the easier task of estimating a level-set compared to estimating a quantile. We also see that ST-GP succeeds in learning TVaR very quickly initially but then its performance plateaus; by $k = 100$ SV-GP achieves better bias and lower $SD(R_k)$. The worse performance of SE-GP for TVaR , especially in terms of very high $SD(R_K)$ confirms the previous discussion.

To check the effect of the batch size Δr, we re-ran ST-GP with $\Delta r = 0.001 \mathcal{N}_1$ (i.e. 9 new inner simulations per round), and observed $\bar{s}$ and RMSE to be 48.37 and 59.22 respectively. These values are statistically indistinguishable from the same ST-GP scheme with $\Delta r = 0.01 \mathcal{N}_1$. In other words, a batch size of $\Delta r = 0.001 \mathcal{N}_1$ (i.e. $K = 1000$ rounds) versus $\Delta r = 0.01 \mathcal{N}_1$ does not provide a significant improvement in this experiment. We found that the smaller batches ended up yielding a procedure that picked the same scenario several times in a row, so that it behaved similarly to a larger batch size anyway.

6.2 Comparing Sequential Schemes

To complement Figure 1 that gives a snapshot of scenarios targeted by different schemes in a fixed dataset, Figure 7 shows the overall budget allocation $r_K^{1:N}$ produced by the ST-GP, SE-GP and

SV-GP at the end of a representative run. For $\mathrm{VaR}_{0.005}$, ST-GP and SE-GP perform similarly with concentration of effort around rank $50 = \alpha N$. For reference, 93%, 73% and 99.1% of the $\mathcal{N}_1$ budget after initialization was allocated to $\{z : f^{(25)} \leq f(z^n) \leq f^{(75)}\}$ for ST-GP, SE-GP and SV-GP respectively, with maximum replication amounts $\max_n r_K^n$ of 3790, 1260 and 990. This highlights the high degree of adaptivity with up to 30% of the budget spent on a single scenario (comparable to the LB benchmark which spends 100% on the quantile scenario).

For TVaR, 95%, 100% and 98.1% of the $\mathcal{N}_1$ budget was spent among $\{z : f^{(1)} \leq f(z^n) \leq f^{(50)}\}$ for ST-GP, SE-GP and SV-GP respectively with maximum allocations of 1890, 4600 and 531. In particular, ST-GP and SV-GP succeed in identifying and sampling (with more or less comparable r_K^n's) from nearly all scenarios in the left tail, while SE-GP leads to a hit-and-miss design as far as the true locations of $f^{(1:50)}$ are concerned. Although all of SE-GP samples were in the true tail, its design tended to be extremely concentrated (only about a dozen non-pilot scenarios added), creating small spatial clusters with a single scenario explored in each cluster.
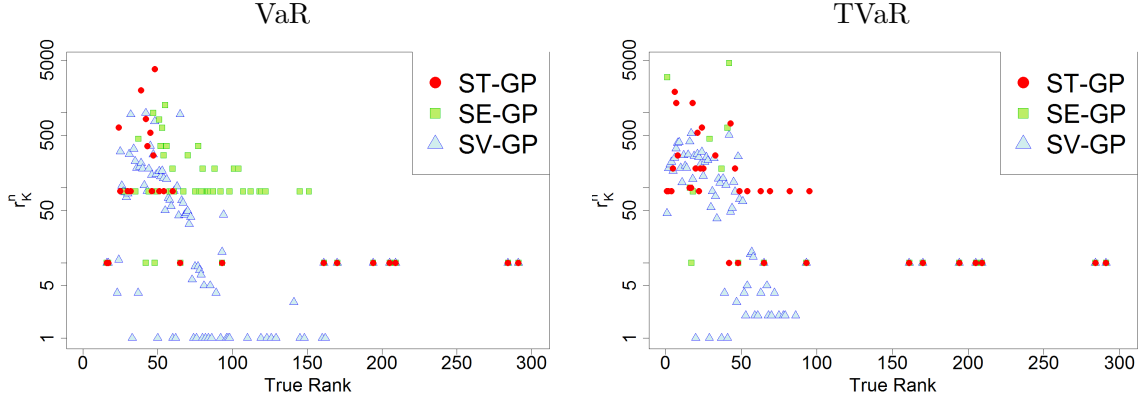

Figure 7: Replication counts r_K^n versus true rank of $f^{1:N}$ after the final stage for sequential methods for the 2-D Black-Scholes case study. $y-$axis is on the log scale. All three schemes share the exact same initialization stage which can be seen via scenarios (especially on the right side of each plot) with $\Delta r_0 = 10$. Otherwise, for ST-GP and SE-GP all r_K^n's are multiples of $\Delta r = 90$. For SV-GP there are no constraints on r_K^n which can be as low as 1.

From a different perspective, Table 3 lists the average total design size $|\mathcal{D}_K|$. We observe that ST-GP and SE-GP only use about 120 scenarios (recall that $N_{init} = 100$ scenarios were already selected during initialization), while SV-GP uses about 300. This is because SV-GP by construction allocates $r_k'^n$ across multiple scenarios, which can be also observed in Figure 7. In particular there are many scenarios that were just "probed" by SV-GP $r_K^n = 1$ (cf. discussion on $\mathcal{Z}^{SV}$ in Section 4.3) but not really used. The resulting larger design size translates into larger computational overhead. Notably, SE-GP is more concentrated than ST-GP and does not appear to explore $\mathcal{Z}$ sufficiently.

Figure 8 illustrates evolution of $s_k(z)$ at $k = 1, 20, 100$ using the SV-GP scheme, for both VaR and TVaR. As k increases, posterior uncertainty $s_k(z^n)$ shrinks; we furthermore see the targeted allocation of inner simulations with $s_k(z^n)$ lowest near the quantile $\mathfrak{q}_\alpha$ (true rank 50) for VaR and throughout the left tail (true rank ≤ 50) for TVaR. There is a clear distinction between the two
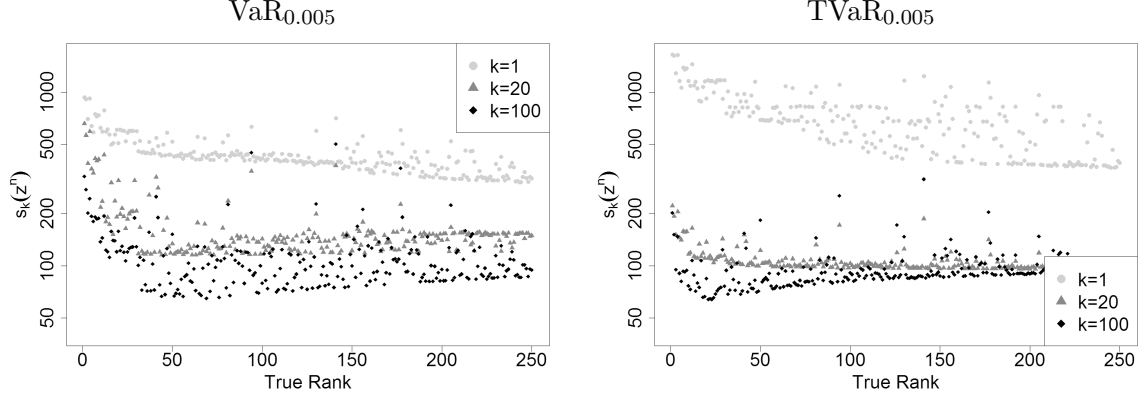
Figure 8: Using SV-GP, posterior GP standard deviation $s_k(z^n)$ at stage $k = 1, 20, 100$ for one run, sorted according to the true rank of $f(z^{1:N})$. $y-$axis is on the log scale. Lower $s_k(z^n)$ indicates higher density of inner simulations in the spatial neighborhood of z^n.

panels, with VaR -0.005 focusing more on rank 50 than rank 1-10, and vice-versa for $TVaR_{0.005}$. This effect is also observed in Figure 9: variance and bias are minimized at $\mathfrak{q}_\alpha$ for VaR$_\alpha$ and in the entire left tail for TVaR$_\alpha$.

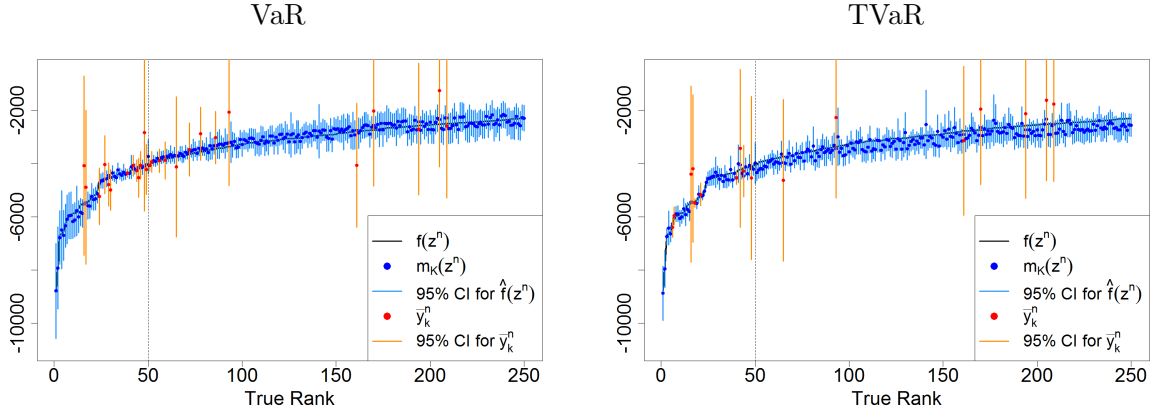## 6.3   Gains from Spatial Modeling

Figure 9: Using ST-GP, output for single run after $K = 100$. Both panels show the true $f(z)$, with point estimates $m_K(z)$ and $\bar{y}_K^n$, and 95% credible/confidence intervals, all sorted according to the true rank of $f(z^{1:N})$. Around each $\bar{y}_K^n$ is a 95% error bar using the hetGP estimate $\tilde{\tau}(z^n)/r_K^n$ for the simulation variance while each $m_K(z^n)$ comes with error bars based on $s_K(z^n)$.

Figure 9 visualizes the spatial features of the GP surrogate at the conclusion of $K = 100$ rounds for a typical run of ST-GP. Recall that our `hetGP` model smoothes both the sample averages $\bar{y}_K^n$ and the sample variances $\hat{\tau}_K^2(z^n)$. This has the double effect of improving the accuracy of learning $f(z^n)$ and lowering the uncertainty of that estimate. Indeed, in the Figure we may see that $m_K(z^n)$ is much closer to $f(z^n)$ relative to the empirical $\bar{y}_K^n$'s and moreover, the GP posterior variance $s_K(z^n)$

is much lower than the empirical $\hat{\tau}_K(z^n)$. We stress that the Figure flattens the underlying 2-D space into a one-dimensional realization, so z's of neighboring rank might be far apart spatially and hence carry quite different $(m_K(z), s_K(z))$ values. In particular, scenarios with large error bars are generally spatially distant from the true quantile contour (left tail, in the TVaR case), whereby the algorithm discovered it was not worth sampling and kept r_K^n low. For scenarios with high r_k^n the two error bars are closer. Large uncertainty and bias are apparent on the *right* side of both panels since the algorithms are explicitly targeting the left tail and hence sacrifice accuracy elsewhere. This feature is also observed on the extreme left for VaR.

To better quantify the improvement in learning R thanks to the spatial borrowing of information across scenarios hinted in Figure 9, we first compare two non-adaptive algorithms: standard nested sampling dubbed U1-SA which works with sample averages $\bar{y}$, versus U1-GP which smoothes them via a GP surrogate. This comparison crystallizes the pure "spatial" effect without any confounding due to the adaptive design. Table 4 reports the results for various simulation budgets $\mathcal{N}$. We see that the use of a spatial model allows a speed-up of about 5x in terms of improved bias and RMSE. For example, U1-SA with $\mathcal{N} = 500,000$ has RMSE of 152.49 which is comparable to the RMSE of U1-GP with just a fifth as many simulations (149.09 for $\mathcal{N} = 100,000$). For really large budgets, the gap narrows since sample averages start to yield decent estimates of $f(z)$. Table 4 also compares the average local empirical variance at the sample quantile $\hat{\tau}(\hat{R}^{HD})$ (used by U1-SA to compute standard errors) against $\bar{s}$, the average GP-based standard error of $\hat{R}_K^{HD}$. We observe an uncertainty reduction of more than 50% for same budgets, again thanks to spatial borrowing.

	U1-SA		U1-GP		BR-SA	
$\mathcal{N}$	$\hat{\tau}_1(\hat{R}_1^{HD})$	RMSE	$\bar{s}$	RMSE	$\hat{\tau}_1(\hat{R}_1^{HD})$	RMSE
$1 \cdot 10^4$	—	6578.12	585.01	2965.05	—	—
$2 \cdot 10^4$	1263.18	3660.92	359.05	1385.44	—	—
$5 \cdot 10^4$	650.62	1569.38	202.97	344.55	135.60	186.82
$1 \cdot 10^5$	410.72	759.87	146.49	148.62	53.50	61.58
$2 \cdot 10^5$	272.09	384.27	107.35	112.63	34.72	43.16
$5 \cdot 10^5$	162.42	163.86	81.26	77.22	19.45	21.58
$1 \cdot 10^6$	112.89	92.44	54.28	66.69	12.98	13.87

Table 4: For the 2-D Black Scholes portfolio case study, average values over 100 macro replications of posterior standard deviation and RMSE for $\hat{R}_K$ using nested MC (U1-SA), the BR-SA algorithm in Broadie et al. [8] and uniform GP surrogate, U1-GP. U1-SA is the standard nested Monte Carlo based on sample averages; U1-GP uses a GP fitted to the simulation output based on single-stage uniform allocation, and BR-SA uses MC sample averages along with a sequential R&S procedure, see Section 5.4. For $\mathcal{N} = 10^4$ the inner simulation budget is $r^n = 1$, not allowing an estimate of $\tau(\mathfrak{q}_\alpha)$ for -SA methods.

From a different angle we can compare the BR-SA method, which does non-uniform allocation of inner simulations, against, say, ST-GP. While BR-SA dramatically outperforms U1-SA (for instance BR-SA with $\mathcal{N} = 5 \cdot 10^4$ is comparable to U1-SA with a budget 10 times larger), it is still far behind the surrogate-based sequential approaches. It takes a 10-20 times larger budget for

BR-SA to begin matching the RMSE's of the methods from the previous section that only employed $\mathcal{N} = 10^4$. So while BR-SA carries essentially no regression/sequential design overhead, the cost in terms of increasing simulation effort is enormous. To conclude, spatial modeling yields an *order-of-magnitude* simulation savings in this example. (We also remark that by its construction, BR-SA first allocates two inner simulations at each scenario and only then enters its adaptive phase. So for $\mathcal{N} \leq 2N$ it reduces to U1-SA; our schemes will produce adaptive allocations even under small simulation budgets.)

6.4 Choice of GP Emulator

To use a GP emulator, one must pick a covariance kernel and also a specific method for fitting the model to the simulation outputs. To investigate the respective impact, we compare use of two different kernel families—Matérn-5/2 versus Gaussian, cf. (14) and (32)—and two different approaches for modeling simulation variance $\tau^2(z)$. Specifically we compare the stochastic kriging approach of using point estimates $\hat{\tau}^2(z^n)$ to estimate $\tau^2(z^n)$ (implemented in `DiceKriging R` package) and the joint response-noise surface approach of `hetGP`. The latter is more complex and hence brings more computational overhead. Table 5 provides final RMSE, as well as bias at several stages when using the ST-GP acquisition function with the above three choices of the GP emulator.

| Approach | Kernel | RMSE | Bias | | | | |
			$k = 1$	$k = 10$	$k = 20$	$k = 50$	$k = 100$
hetGP	Matérn-5/2	57.52	166.065	113.925	97.751	37.158	28.066
hetGP	Gaussian	68.06	91.475	104.427	74.874	52.716	48.249
SK	Matérn-5/2	69.31	914.728	206.267	113.716	69.821	48.670

Table 5: For the 2-D Black Scholes portfolio case study, average RMSE of $\hat{R}_K$ across different GP models/kernel families. We also report average bias bias($\hat{R}_k$) across a selection of intermediate stages $k = 1, 10, 20, 50, 100$. All methods use the ST-GP rule and are based on 100 macro-replications.

We observe that using the empirical variance estimates $\hat{\tau}^2(z^n)$ (SK) generates glaringly high biases at initial stages. In turn, the `hetGP` model is able to de-bias efficiently. Although bias falls rapidly as the algorithm progresses, the SK emulator still has a 50% higher bias (and 20% higher RMSE) at the $k = 100$ stage. Because in practice one does not know when the bias has been effectively alleviated, this is a significant shortcoming of working with an SK emulator and highlights the importance of properly modeling the noise surface $\tau(\cdot)$. In comparing kernels, the Gaussian kernel shows slightly higher RMSE and bias compared to Matérn-5/2, though not by a significant margin. This matches the folklore view that Matérn is the default choice for the kernel family.

7 Case Study: Life Annuities under Stochastic Interest Rate and Mortality

To check if the relative performance of each method remains the same in a more elaborate setting, we move to a case study in higher dimensions with a more complicated payoff function. Despite increased complexity we maintain the same budget $\mathcal{N} = 10^4$ which is expected to magnify the relative differences across proposed methods.

The setup in this section considers an annuitant who enters into a contract to begin collecting payments in S years, whence the payments continue annually until death of the individual. (In practice some cutoff age x_u is set for the final payment.) Regulations require analysis of quantiles of the $T = 1$-year value of this contract to the insurer. The major drivers of portfolio loss are interest rate risk (low interest rates increasing the present value of annuity payments) and mortality risk (increased longevity raising the value of the annuity). These factors are captured by 6-D stochastic state: a three-factor model for interest rates and a three-factor stochastic mortality model.

We wish to find the present value of this annuity at horizon $T \leq S$, before payments begin at S. To this end, let $\mathcal{T}(x)$ be the remaining random lifetime of an individual aged x today. The annuity payment at t is predicated on $\mathcal{T}(x) > t$ whose likelihood $P(t, x)$ we represent as

$$P(t, x) = Pr(\mathcal{T}(x) > t) = 1 - \sum_{u=0}^{t-1} Pr(u < \mathcal{T}(x) \leq u + 1)$$

$$\dot{=} 1 - \sum_{u=0}^{t-1} q(u, x + u), \tag{36}$$

where $q(u, x + u)$ is the mortality rate, i.e. the probability of an individual aged $x + u$ dying in calendar year u (alternatively interpreted as individual aged x today dying between ages $x + u$ and $x + u + 1$). Integrating out idiosyncratic mortality experience (denoted by $Pr(\cdot)$ in (36)), we focus on systematic factors that affect future evolution of mortality. Thus, we view $q(u, x + u)$ as a random variable that is driven by stochastic mortality factors (Z_t), i.e. $q(Z_t; t, x)$. This implies that the survival probability $P(Z_{[T,t]}; t, x)$ in (36) depends on the whole *path* $Z_{[T,t]}$ between the horizon T and the summed years t.

Let β_t be the instantaneous interest rate at time t, which also depends on (other) components of (Z_t). The net present value of the annuity at T (conditioning on $\mathcal{T}(x) > T$ and assuming that the longevity and interest rate risks are independent) is

$$f(z) = -\mathbb{E}\left[\sum_{t=S}^{x_u-x} e^{-\int_T^t \beta_u du} P(Z_{[T,t]}; t, x) \,\middle|\, Z_T = z, \mathcal{T}(x) > T\right]. \tag{37}$$

For survival probabilities we choose the following model, known as (M7) in Cairns et al. [9]:

$$\text{logit}\, q(Z_t; t, x) = \kappa_t^1 + \kappa_t^2(x - \bar{x}) + \kappa_t^3\left((x - \bar{x})^2 - \hat{\sigma}_x^2\right) + \gamma_{t-x}. \tag{38}$$

In (38), the stochastic drivers are $\kappa_t^j, j = 1, 2, 3$ as well as γ_{t-x}; the rest are fixed parameters. Specifically, $\bar{x}$ is the average age the model is fit to, and $\hat{\sigma}_x^2$ is the mean value over fitted ages x

of $(x - \bar{x})^2$, which are interpreted as *age* effects. The model allows a plugin for age x, while the stochastic factors are the *period* effects κ_t^i which capture mortality evolution over calendar year, and γ_{t-x} which is the *cohort* effect. These constitute discrete-time ARIMA models, following the common choice that each κ^i is a random walk with drift. Typically, $\gamma_.$ is fitted as an ARMA model.

For reproducibility, we use the R package StMoMo [37] which contains tools for fitting and simulating (38). Namely, we utilize the England & Wales (E&W) mortality contained in the data and previously studied in Cairns et al. [9]. Because the expression (36) for $P(Z_{[T,t]}; t, x)$ has Equation (38) evaluated at $(u, x + u)$ for age/year, it results in a fixed cohort effect γ_x fitted from historic values of the ARMA process.

The interest rate dynamics for (β_t) are from [13], defined through a three-factor Cox-Ingersoll-Ross model with stochastic volatility ζ_t and stochastic drift α_t

$$
\begin{aligned}
d\beta_t &= (\bar{\beta} - \alpha_t)dt + \sqrt{\beta_t}\zeta_t \, dW_t^\beta, \\
d\alpha_t &= (\bar{\alpha} - \alpha_t)dt + \sqrt{\alpha_t}\zeta_t \, dW_t^\alpha, \\
d\zeta_t &= (\bar{\zeta} - \zeta_t)dt + \sqrt{\zeta_t}\varphi \, dW_t^\zeta,
\end{aligned}
\tag{39}
$$

where W^β, W^α and W^ζ are independent standard Brownian motions.

Overall, this implies a 6-D Markov state process $Z_t = (\beta_t, \alpha_t, \zeta_t, \kappa_t^1, \kappa_t^2, \kappa_t^3)$. Thus, the procedure to obtain payoff realizations $Y^i(z^{n,i})$ is to first simulate paths $(z_t^{n,i})_{t \geq T}$ from the distribution $(Z_t)_{t \geq T} | Z_T = z^n$, plug-in the realizations for the $(\kappa_t^j)_{t \geq T}$, $j = 1, 2, 3$ into (38) to evaluate (36), and finally use these survival probabilities as well as the simulated $(\beta_t)_{t \geq T}$ to compute

$$
Y^i(z^{n,i}) = \sum_{t=S}^{x_u - x} e^{-\int_T^t \beta_u^{n,i} du} P(z_{[T,t]}^{n,i}; t, x).
\tag{40}
$$

Note that the κ's are discrete-time, while the interest rate model uses continuous t. For the latter, we use a simple forward Euler method with discretization $\Delta t = 0.1$. In all, one cashflow $Y(\cdot)$ takes approximately 0.01115 seconds to evaluate (i.e. about 2 minutes for the 10^4 overall inner simulations), while it takes 0.000513 seconds to evaluate for the first case study. Such a computationally expensive simulator is more representative of real-life simulation engines and reduces the impact of overhead costs in emulator fitting, selection, and prediction.

7.1 Results

For the remainder of this section, we consider the horizon $T = 1$ and analyze $f(Z_1)$ in Equation (37), the net present value of the life annuity one year into the future. We take $x = 55$ and annuity start date of $S = 10$, when the individual retires at age $x = 65$. As before, we simulate Z_1 via Algorithm 1 in Appendix B to determine a scenario set $\mathcal{Z}$ with $N = 10^4$. The interest rate parameters in Equation (39) are $\bar{\beta} = 0.04, \bar{\alpha} = 0.04, \bar{\zeta} = 0.02, \phi = 0.05$, with the E&W mortality model fitted over the age range $x \in [55, 89]$ using the StMoMo package [37]. As in the previous case study, the GP hyperparameters are refitted after stages $10, 20, \ldots, 100$, and after each round for A3-GP and U2-GP. For the mean function, we fit the constant $\mu(z) \equiv \beta_0$.

Under the setup (37), there is no closed form evaluation for $f(z)$, so we obtain a benchmark R_B through simulation—this value is determined by running the SR-GP approach with $\mathcal{N} = 2 \cdot 10^7$

using $K = 200$ rounds (so that $\Delta r_k = 1 \cdot 10^5$; we also take a very conservative $L = 1, U = 200$). The result is $R_B^{\text{VaR}} = -16.0498$ and $R_B^{\text{TVaR}} = -16.3739$, with estimator standard deviations of $s(R_B^{\text{VaR}}) = 0.00202$ and $s(R_B^{\text{TVaR}}) = 0.00188$.

We repeat the methods of Section 6, performing 100 macro replications with fixed $\mathcal{Z}$ for both $\text{VaR}_{0.005}$ and $\text{TVaR}_{0.005}$. The boxplots for bias and posterior uncertainty are reported in Figure 10, and the numeric values for estimator standard deviation, RMSE, and final design sizes in Table 6.

	$\text{VaR}_{0.005}$				$\text{TVaR}_{0.005}$				
	$SD(\hat{R}_K^{HD})$	$\overline{s}$	RMSE	$\lvert\mathcal{D}_K\rvert$	$SD(\hat{R}_K)$	$\overline{s}$	RMSE	$\lvert\mathcal{D}_K\rvert$	Time
ST-GP	0.0394	0.0455	0.0403	151.83	0.0461	0.0404	0.0472	147.10	330
SE-GP	0.0427	0.0493	0.0459	143.27	0.2853	0.2717	0.2850	101.62	295
SV-GP	0.0382	0.0406	0.0380	497.81	0.0408	0.0393	0.0407	254.35	403
SR-GP-1	0.0467	0.0470	0.0485	135.03	0.0430	0.0402	0.0430	184.73	219
SR-GP-2	0.0391	0.0437	0.0434	217.36	0.0447	0.0431	0.0450	224.96	198
A3-GP	0.0434	0.0497	0.0436	298.15	0.0464	0.0490	0.0461	298.55	115
U2-GP	0.0598	0.0598	0.0596	194.03	0.0684	0.0601	0.0689	195.81	112
U1-GP	0.5020	0.4156	0.5853	10^4	0.4940	0.4705	0.6709	10^4	177

Table 6: Results for the 6-D life annuity case study based on 100 macro-replications. We report sample standard deviation of $\hat{R}_K^{[1:100]}$, average GP posterior standard deviation $\overline{s}$, and RMSE of $\hat{R}_K$, as well as average final design size for each approach. The last column reports the running time in seconds. Description of the methods is in Table 1.

The results only slightly differ from the first case study in terms of relative performance. For $\text{VaR}_{0.005}$, SV-GP again has the lowest RMSE value, although ST-GP is only 6% worse. Interestingly, A3-GP performs as well as SR-GP-2. Since A3-GP allocates Δr_1 among 200 locations, this suggests that the bandwidth $L = 26, U = 75$ for SR-GP-2 may be too tight for this more complex setup, especially since this time it only slightly outperforms the aggressive SR-GP-1. As before, U1-GP fails (RMSE is more than 15x higher than for SV-GP), and while U2-GP offers a large improvement, it still has a much higher RMSE compared to the other methods. The boxplots in Figure 10 show that the most stable uncertainty quantification (tight distribution of $s(\hat{R}_K^{HD})$ and close to the actual $SD(\hat{R}_K)$) is provided by ST-GP, SV-GP and SR-GP-2. Like in the first case study, SE-GP leads to more across-run dispersion in both $\hat{R}_K$ and its reported standard error.

The TVaR results are similar; we comment on a few noticeable differences. SV-GP still has the lowest RMSE and the lowest $SD(\hat{R}_K)$. At the other end, SE-GP shows the same issues with its acquisition function and unstable $s(\hat{R}_K)$. The SR-GP methods do second best (although their uncertainty quantification is significantly less reliable with unstable $s(\hat{R}_K)$), with A3-GP and ST-GP close behind. We attribute the relative "win" of SV-GP to its more diffuse allocation, i.e. larger $\lvert\mathcal{D}_K\rvert$, which indicates that estimating TVaR in a more complex example requires wider casting of the net, rather than narrowly targeting the tail. Overall, this case study confirms the performance gains of the GP-based sequential methods even in a higher-dimensional setting.
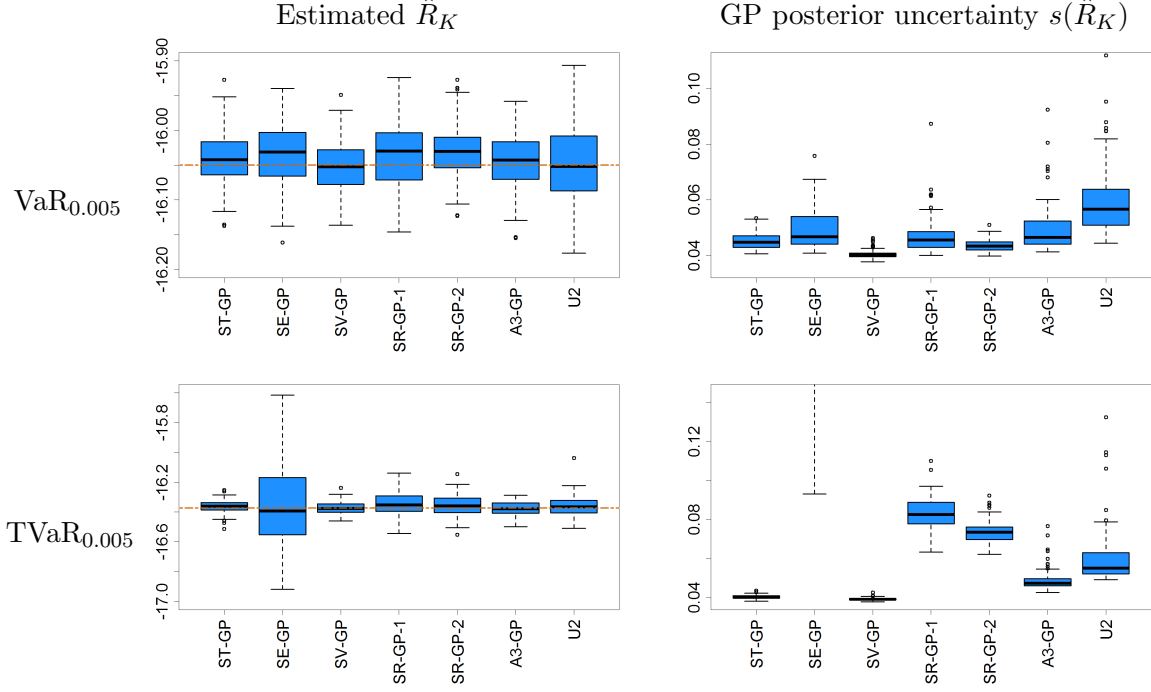
Figure 10: For the life annuity case study, boxplots of final $\hat{R}_K$ estimates (left) and $s(\hat{R}_K)$ (right) over 100 macro replications for each approach. Top row: Value-at-Risk; bottom row: TVaR. The orange dotted lines on the left are the reference values of the risk measure.

8 Conclusion

We provided the first comprehensive examination of sequential methods for estimating VaR_α and TVaR_α using Gaussian process emulators and nested simulations over a fixed set of outer scenarios. Existing approaches in the simulation literature fall short for this application, whether it be assuming the level (VaR_α) is known [30, 4, 15] and/or non-noisy outputs [26], using a non sequential algorithm [27], or not using a spatial model for smoothing [8].

We document a *two orders-of-magnitude* savings relative to plain nested Monte Carlo, with about 10x savings thanks to adaptive allocation and 5x savings due to spatial modeling. These gains are strongest for small simulation budgets $\mathcal{N} \approx N$ where existing approaches essentially fail or are not applicable. As a further advantage, employing a GP emulator also leads to essentially unbiased risk estimators (in contrast to biased unsmoothed versions) and provides a reliable estimate of standard error. We also took advantage of the newly introduced heteroskedastic GP framework in `hetGP` [6]. The improved estimation of the noise surface $\tau^2(\cdot)$ slashes the bias in $\hat{f}$ when replicate counts r_k's are low, and yields a more robust estimator for $\hat{R}_k$ throughout.

All fully sequential methods showed significant improvement over both the simple two-stage approach and three-stage method of [27]. The major downside to our combination of GP emulation and sequential design is the regression/prediction overhead, however this is reduced significantly by batching and carefully choosing candidate sets $\mathcal{Z}_k^{cand}$. In addition, the overall time spent computing

acquisition functions, fitting and predicting becomes negligible in more realistic examples where calls to $Y(\cdot)$ are expensive.

We have identified two top-performing approaches. SV-GP, the fully sequential adaptation of [27], did best in terms of RMSE in both case studies. SV-GP also led to the smallest variation in the outputted posterior uncertainty $s_K(\hat{R}_K)$, allowing this value to be a reliable proxy for the true sampling standard deviation. ST-GP performed nearly as well, and in fact appeared to do best for the first few rounds, cf. the fan plots of Figure 6. The implementation of SV-GP involves a separate auxiliary optimization problem and brings more overhead and coding complexity. In contrast the criterion of ST-GP is straightforward and also keeps the emulator model size to a minimum (in terms of $|\mathcal{D}_k|$ which is the key determinant for GP overhead). An even simpler/faster scheme is the rank-based SR-GP that also performed well, but its results depended heavily on the choices of L and U, which in turn were varying among case studies and VaR$_\alpha$ versus TVaR$_\alpha$. This is a further advantage of SV-GP and ST-GP that require no user-specified parameters. The mixed performance of SE-GP offers a cautionary tale that designing a good acquisition function is important for overall performance. Our take-away is that ST-GP is an excellent choice when budget $\mathcal{N}$ is really limited, while for larger budgets a scheme like SV-GP is best.

There are multiple ways to move further. The best-performing SV-GP could be further embellished through fine-tuning the weights $\mathbf{w}$ that are used during the respective look-ahead variance optimization. For example, the related three-stage approach in [27] uses an empirical weight estimate obtained through simulation in order to determine weights for the TVaR estimator. Modifying the optimization criterion to avoid the requirement that $r_k^n \geq 1$ would remove the scenario probing and drastically shrink SV-GP's design size $|\mathcal{D}_k|$. Alternatively, adaptive selection of Δr_k would reduce emulation overhead.

Conversely, one could extend the ST-GP or SE-GP approaches to adaptively optimize over both z^{k+1} and Δr_k, i.e. to internalize the batching into the design construction (whereas currently the respective acquisition functions are technically 1-step lookahead). Theoretical analysis regarding various convergence *rates* of $\hat{R}_K$-bias would also be welcome, although it is already extremely challenging for the simpler problem of contour finding with a known threshold L.

The initialization stage also warrants further analysis. Recall that we used a fixed budget of $\Delta r_0 = 0.1\mathcal{N}$, spent through N_{init} representative scenarios determined via a space-filling scheme. In principle, Δr_0 should be taken as small as possible, letting the algorithm itself allocate the rest of the simulations, but in practice a decent starting point for the GP hyperparameters is crucial. Better rules/heuristics for the choice of Δr_0 remain to be considered. Also, rather than space-filling the given $\mathcal{Z}$, it could be desirable to focus further on the respective extreme regions, in the sense of geometrically extremal points of $\mathcal{Z}$, see e.g. the use of statistical depth functions in [11].

Last but not least, a variety of adjustments could be made to the GP emulators underlying our framework. Recall that the surrogate makes a number of assumptions, such as Gaussian observation noise and spatial stationarity in the covariance structure. These are likely to be violated in practice; if model mis-specification is a concern, one could entertain numerous alternatives that constitute an active area of research within the GP ecosystem—GP-GLM (e.g. with t-distributed noise), treed GPs and local GPs.

References

[1] BRUCE ANKENMAN, BARRY L NELSON, AND JEREMY STAUM, *Stochastic kriging for simulation metamodeling*, Operations Research, 58 (2010), pp. 371–382.

[2] BASEL COMMITTEE, *Fundamental review of the trading book: A revised market risk framework*, consultative document, October ed., 2013.

[3] DANIEL BAUER, ANDREAS REUSS, AND DANIELA SINGER, *On the calculation of the solvency capital requirement based on nested simulations*, ASTIN Bulletin: The Journal of the IAA, 42 (2012), pp. 453–499.

[4] JULIEN BECT, DAVID GINSBOURGER, LING LI, VICTOR PICHENY, AND EMMANUEL VAZQUEZ, *Sequential design of computer experiments for the estimation of a probability of failure*, Statistics and Computing, 22 (2012), pp. 773–793.

[5] MICKAEL BINOIS AND ROBERT GRAMACY, *hetGP: Heteroskedastic Gaussian Process Modeling and Design under Replication*, 2017. CRAN package v.1.0.0.

[6] MICKAEL BINOIS, ROBERT B GRAMACY, AND MICHAEL LUDKOVSKI, *Practical heteroskedastic Gaussian process modeling for large simulation experiments*, tech. report, arXiv preprint arXiv:1611.05902, 2016.

[7] KURT M BRETTHAUER, ANTHONY ROSS, AND BALA SHETTY, *Nonlinear integer programming for optimal allocation in stratified sampling*, European Journal of Operational Research, 116 (1999), pp. 667–680.

[8] MARK BROADIE, YIPING DU, AND CIAMAC C MOALLEMI, *Efficient risk estimation via nested sequential simulation*, Management Science, 57 (2011), pp. 1172–1194.

[9] ANDREW JG CAIRNS, DAVID BLAKE, KEVIN DOWD, GUY D COUGHLAN, DAVID EPSTEIN, AND MARWA KHALAF-ALLAH, *Mortality density forecasts: An analysis of six stochastic mortality models*, Insurance: Mathematics and Economics, 48 (2011), pp. 355–367.

[10] TONY F CHAN, GENE HOWARD GOLUB, AND RANDALL J LEVEQUE, *Updating formulae and a pairwise algorithm for computing sample variances*, in COMPSTAT 1982 5th Symposium held at Toulouse 1982, Springer, 1982, pp. 30–41.

[11] MATTHIEU CHAUVIGNY, LAURENT DEVINEAU, STÉPHANE LOISEL, AND VÉRONIQUE MAUME-DESCHAMPS, *Fast remote but not extreme quantiles with multiple factors: applications to Solvency II and enterprise risk management*, European Actuarial Journal, 1 (2011), pp. 131–157.

[12] CHUN-HUNG CHEN, DONGHAI HE, MICHAEL FU, AND LOO HAY LEE, *Efficient simulation budget allocation for selecting an optimal subset*, INFORMS Journal on Computing, 20 (2008), pp. 579–595.

[13] Lin Chen, *Stochastic mean and stochastic volatility: a three-factor model of the term structure of interest rates and its applications in derivatives pricing and risk management*, in Interest Rate Dynamics, Derivatives Pricing, and Risk Management, vol. 435 of Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, 1996, pp. 1–36.

[14] Xi Chen, Barry L Nelson, and Kyoung-Kuk Kim, *Stochastic kriging for conditional value-at-risk and its sensitivities*, in Proceedings of the 2012 Winter Simulation Conference (WSC), IEEE, 2012, pp. 1–12.

[15] Clément Chevalier, Julien Bect, David Ginsbourger, Emmanuel Vazquez, Victor Picheny, and Yann Richet, *Fast parallel kriging-based stepwise uncertainty reduction with application to the identification of an excursion set*, Technometrics, 56 (2014), pp. 455–465.

[16] Clément Chevalier, Victor Picheny, and David Ginsbourger, *Kriginv: An efficient and user-friendly implementation of batch-sequential inversion strategies based on kriging*, Computational Statistics & Data Analysis, 71 (2014), pp. 1021–1034.

[17] Marcus C Christiansen and Andreas Niemeyer, *Fundamental definition of the solvency capital requirement in Solvency II*, ASTIN Bulletin: The Journal of the IAA, 44 (2014), pp. 501–533.

[18] Alexander Forrester, Andras Sobester, and Andy Keane, *Engineering design via surrogate modelling: a practical guide*, John Wiley & Sons, 2008.

[19] Peter I Frazier, *A fully sequential elimination procedure for indifference-zone ranking and selection with tight bounds on probability of correct selection*, Operations Research, 62 (2014), pp. 926–942.

[20] Gene H Golub and Charles F Van Loan, *Matrix computations*, vol. 3, JHU Press, 2012.

[21] Michael B Gordy and Sandeep Juneja, *Nested simulation in portfolio risk measurement*, Management Science, 56 (2010), pp. 1833–1848.

[22] Robert B Gramacy and Herbert KH Lee, *Gaussian processes and limiting linear models*, Computational Statistics & Data Analysis, 53 (2008), pp. 123–136.

[23] Frank E Harrell and CE Davis, *A new distribution-free quantile estimator*, Biometrika, 69 (1982), pp. 635–640.

[24] Bogumil Kamiński, *A method for the updating of stochastic kriging metamodels*, European Journal of Operational Research, 247 (2015), pp. 859–866.

[25] Joseph Hyun Tae Kim and Mary R Hardy, *Quantifying and correcting the bias in estimated risk measures*, Astin Bulletin, 37 (2007), pp. 365–386.

[26] T Labopin-Richard and V Picheny, *Sequential design of experiments for estimating percentiles of black-box functions*, Statistica Sinica, to Appear (2016). arXiv preprint arXiv:1605.05524.

[27] Ming Liu and Jeremy Staum, *Stochastic kriging for efficient nested simulation of expected shortfall*, Journal of Risk, 12 (2010), pp. 3–27.

[28] Amandine Marrel, Bertrand Iooss, François Van Dorpe, and Elena Volkova, *An efficient methodology for modeling complex computer codes with Gaussian processes*, Computational Statistics & Data Analysis, 52 (2008), pp. 4731–4744.

[29] Jeremy Oakley, *Estimating percentiles of uncertain computer code outputs*, Journal of the Royal Statistical Society: Series C (Applied Statistics), 53 (2004), pp. 83–93.

[30] Victor Picheny, David Ginsbourger, Olivier Roustant, Raphael T Haftka, and Nam-Ho Kim, *Adaptive designs of experiments for accurate approximation of a target region*, Journal of Mechanical Design, 132 (2010), p. 071008.

[31] Carl Edward Rasmussen and Christopher K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[32] James Risk and Michael Ludkovski, *Statistical emulators for pricing and hedging longevity risk products*, Insurance: Mathematics and Economics, 68 (2016), pp. 45–60.

[33] Olivier Roustant, David Ginsbourger, and Yves Deville, *DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by kriging-based metamodeling and optimization*, Journal of Statistical Software, 51 (2012), pp. 1–55.

[34] Thomas J Santner, Brian J Williams, and William I Notz, *The design and analysis of computer experiments*, Springer Science & Business Media, 2013.

[35] Michael E Sfakianakis and Dimitris G Verginis, *A new family of nonparametric quantile estimators*, Communications in Statistics: Simulation and Computation, 37 (2008), pp. 337–345.

[36] Simon J Sheather and James Stephen Marron, *Kernel quantile estimators*, Journal of the American Statistical Association, 85 (1990), pp. 410–416.

[37] Andres M. Villegas, Pietro Millossovich, and Vladimir K. Kaishev, *StMoMo: An R Package for Stochastic Mortality Modelling*, 2017. R package version 0.4.0.

A   Variance Minimization Calculations

Lemma 1. *We have the following approximation that improves as each r_{k+1}^n increases:*

$$(\mathbf{C} + \mathbf{\Delta}_{k+1}^{cand})^{-1} \approx (\mathbf{C} + \mathbf{\Delta}_k)^{-1} + (\mathbf{C} + \mathbf{\Delta}_k)^{-1}(\mathbf{\Delta}_k - \mathbf{\Delta}_{k+1}^{cand})(\mathbf{C} + \mathbf{\Delta}_k)^{-1}. \tag{41}$$

Proof. Recall that $\mathbf{\Delta}_{k+1}^{cand}$ is diagonal with entries $\tau^2(z^n)/(r_k^n + r_k'^n)$ and $\mathbf{\Delta}_k$ is diagonal with entries $\tau^2(z^n)/r_k^n$. We re-write

$$\mathbf{\Delta}_{k+1}^{cand} = \mathbf{\Delta}_k + \mathbf{B}_{k+1}(-\mathbf{I})\mathbf{B}_{k+1},$$

where $\mathbf{B}_{k+1}$ is a diagonal matrix with elements $b_{nn} \doteq \tau(z^n)\sqrt{\left(\frac{1}{r_k^n} - \frac{1}{r_k^n + r_k'^n}\right)}$ and $\mathbf{I}$ is the identity matrix. By the Woodbury matrix inversion formula [20] we then obtain

$$
\begin{aligned}
(\mathbf{C} + \boldsymbol{\Delta}_{k+1}^{cand})^{-1} &= (\mathbf{C} + \boldsymbol{\Delta}_k + \mathbf{B}_{k+1}(-\mathbf{I})\mathbf{B}_{k+1})^{-1} \\
&= (\mathbf{C} + \boldsymbol{\Delta}_k)^{-1} \\
&\quad - (\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{B}_{k+1}\left(\mathbf{B}_{k+1}(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{B}_{k+1} - \mathbf{I}\right)^{-1}\mathbf{B}_{k+1}(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}.
\end{aligned}
\tag{42}
$$

When r_{k+1}^n is large, both $\mathbf{B}_{k+1}$ and $\boldsymbol{\Delta}_k$ have relatively small entries, so that $\mathbf{B}_{k+1}(\mathbf{C}+\boldsymbol{\Delta}_k)^{-1}\mathbf{B}_{k+1} \approx \mathbf{0}$. Therefore, the middle matrix inverse in (42) is $\approx (-\mathbf{I})$ and

$$
(\mathbf{C} + \boldsymbol{\Delta}_{k+1}^{cand})^{-1} \approx (\mathbf{C} + \boldsymbol{\Delta}_k)^{-1} + (\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{B}_{k+1}\mathbf{B}_{k+1}(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}.
\tag{43}
$$

Plugging in $\mathbf{B}_{k+1}^2 = \boldsymbol{\Delta}_k - \boldsymbol{\Delta}_{k+1}^{cand}$, we have (41).

Returning to (27), the optimization of the RHS is over $r_k'^n$'s which appear only inside $\boldsymbol{\Delta}_{k+1}^{cand}$. Plugging in (41) and dropping all terms that do not depend on r'^n, we obtain that minimizing (27) is equivalent to *maximizing*

$$
\hat{\mathbf{w}}_k \mathbf{C}(\mathbf{C} + \boldsymbol{\Delta}_{k+1}^{cand})^{-1}\mathbf{C}\hat{\mathbf{w}}_k^T = \hat{\mathbf{w}}_k \mathbf{C}(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}(\boldsymbol{\Delta}_k - \boldsymbol{\Delta}_{k+1}^{cand})(\mathbf{C} + \boldsymbol{\Delta}_k)^{-1}\mathbf{C}\hat{\mathbf{w}}_k^T.
$$

After dropping the term with $\boldsymbol{\Delta}_k$ (which is still independent of r'^n) this reduces to *minimizing* (28). □

B   Generating Pilot Scenarios

To construct N_{init} pilot scenarios for the stage-0 design $\mathcal{D}_0$ we first standardize the scenario set $\mathbf{z} = (z^1, \ldots, z^N)$, via

$$
z_{std}^n \doteq \frac{z^n - \bar{\mu}_z^n}{\sigma_z^n},
\tag{44}
$$

where $\bar{\mu}_z^n$ and σ_z^n are the coordinate-wise sample mean and standard deviation of $\mathbf{z}$. We next consider a parameter d_0 that determines the desired density of pilot scenarios. To pick the latter, we randomly permute the order of $\mathbf{z}$ and sequentially add new design locations whenever they are at least d_0 (Euclidean) distance away from the current sites, continuing until all z^n's are tried. While the resulting design size $\leq N_{init}$ depends on the random permutation, we found that practically it is only slightly affected by the randomization in the order of $\mathbf{z}$. If a specific size N_{init} of $\mathcal{D}_0$ is desired, one approach is to start with relatively large d_0 ($d_0 = 10\sqrt{d}N_{init}^{-1}$ is a good start), and sequentially reduce its value if the algorithm cannot find enough locations satisfying the distance requirement.

Input: z, N_{init}, d_0

    Randomize the order of **z**;

    Compute $\mathbf{z}_{std}$ according to (44);

    Initialize $I \leftarrow \{1\}$; $j \leftarrow 2$;

    while $\mathrm{card}(I) < N_{init}$ **do**

        if $\min_{i \in I} \| z_{std}^{j}, z_{std}^{i} \|_2 \geq d_0$ **then**

            $I \leftarrow I \cup \{j\}$ // Add z^j

        end if

        $j \leftarrow j + 1$;

    end while

    Return pilot set $\mathcal{D}_0 = \{z^n : n \in I\}$ and $N_{init} = \mathrm{card}(I)$

                Algorithm 1: A space-filling design for initializing the emulator