Received: 9 November 2021 | Revised: 3 November 2022 | Accepted: 14 November 2022

DOI: 10.1002/nla.2481

SPECIAL ISSUE PAPER

WILEY

TTRISK: Tensor train decomposition algorithm for risk averse optimization

Harbir Antil[1] | **Sergey Dolgov**[2] | **Akwum Onwunta**[3]

[1]The Center for Mathematics and Artificial Intelligence (CMAI) and Department of Mathematical Sciences, George Mason University, Fairfax, Virginia, USA

[2]Department of Mathematical Sciences, University of Bath, Bath, UK

[3]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, Pennsylvania, USA

Correspondence
Sergey Dolgov, Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK.
Email: s.dolgov@bath.ac.uk

Funding information
Air Force Office of Scientific Research, Grant/Award Numbers: FA9550-19-1-0036, FA9550-22-1-0248; Engineering and Physical Sciences Research Council, Grant/Award Number: EP/T031255/1, EP/V04771X/1; National Science Foundation, Grant/Award Numbers: DMS-2110263, DMS-1913004

Abstract

This article develops a new algorithm named TTRISK to solve high-dimensional risk-averse optimization problems governed by differential equations (ODEs and/or partial differential equations [PDEs]) under uncertainty. As an example, we focus on the so-called Conditional Value at Risk (CVaR), but the approach is equally applicable to other coherent risk measures. Both the full and reduced space formulations are considered. The algorithm is based on low rank tensor approximations of random fields discretized using stochastic collocation. To avoid nonsmoothness of the objective function underpinning the CVaR, we propose an adaptive strategy to select the width parameter of the smoothed CVaR to balance the smoothing and tensor approximation errors. Moreover, unbiased Monte Carlo CVaR estimate can be computed by using the smoothed CVaR as a control variate. To accelerate the computations, we introduce an efficient preconditioner for the Karush–Kuhn–Tucker (KKT) system in the full space formulation.The numerical experiments demonstrate that the proposed method enables accurate CVaR optimization constrained by large-scale discretized systems. In particular, the first example consists of an elliptic PDE with random coefficients as constraints. The second example is motivated by a realistic application to devise a lockdown plan for United Kingdom under COVID-19. The results indicate that the risk-averse framework is feasible with the tensor approximations under tens of random variables.

KEYWORDS

CVaR, full space, preconditioner, reduced space, risk measures, tensor train, TTRISK

1 | INTRODUCTION

Uncertainty is ubiquitous in science and engineering applications. It may arise due to noisy measurements, unknown parameters, or unverifiable modeling assumptions. Examples include, infectious disease models for COVID-19,[1] partial differential equations (PDEs) with random coefficients, boundary conditions or right-hand sides.[2-5] The control or design optimization problems constrained by such systems must produce controls or optimal designs which are resilient to this uncertainty. To tackle this, recently in References 5-8, the authors have created risk-averse optimization frameworks targeting engineering applications.

The goal of this paper is to introduce a new algorithm TTRISK which uses a Tensor Train (TT) decomposition to solve risk averse optimization problems constrained by differential equations (ODEs and/or PDEs). Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a complete probability space. Let U, Y be real reflexive Banach spaces, and let Z be a real Banach space. Here Y denotes the deterministic state space, U is the space of optimization variables (control or designs etc.) and Z is the differential equation residual space. Let $U_{ad} \subseteq U$ be a closed convex subset and let $c : Y \times U_{ad} \times \Omega \to Z$ denote, for example, a partial differential operator, then consider the equality constraint

$$c(y, u; \omega) = 0, \quad \text{in } Z, \quad \text{a.a. } \omega \in \Omega,$$

where a.a. indicates "almost all" with respect to a probability measure $\mathbb{P}$. The goal of this article is to consider optimization problems of the form

$$\min_{u \in U_{ad}} \mathcal{R}[\mathcal{J}(y, u; \omega)] + \alpha \mathcal{P}(u) \text{ subject to } c(y, u; \omega) = 0, \quad \text{in } Z, \quad \text{a.a. } \omega \in \Omega, \tag{1}$$

where $u \in U_{ad}$ is the deterministic control and $y \in Y$ is the state, $\mathcal{P}$ is the cost of the control, $\alpha \geq 0$ is the regularization parameter, $\mathcal{J}$ is the uncertain variable objective function and $\mathcal{R}$ is the risk-measure functional which maps random variables to extended real numbers.

We assume that $\mathcal{R}$ is based on expectation, that is,

$$\mathcal{R}[X] = \inf_{t \in \mathcal{T}} \mathcal{R}_t[X], \quad \text{where} \quad \mathcal{R}_t[X] := \mathbb{E}[f(X, t)], \tag{2}$$

$f : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$ and $\mathcal{T} \subseteq \mathbb{R}^N$, with $N \in \mathbb{N}$, is a closed convex set. The problem class (2) consists of a large number of risk-measures that are of practical interest. In particular, it includes the *coherent risk measures*, which are sub-additive, monotonic, translation equivariant and positive homogeneous.[9,10] Notice that subadditivity and positive homogeneity implies convexity. These risk measures have several advantages, for instance, they preserve desirable properties of the original objective function such as convexity. In addition, in engineering or in finance applications, tail-probability events may be rare but critical if they lead to failure of a system. It is therefore essential to minimize the risk of failure, that is, $\mathcal{R}$, and obtain controls u which are resilient to uncertainty in the system.

A typical example of coherent risk measure $\mathcal{R}$ is the conditional value-at-risk (CVaR_β), where f in (2) is given by

$$f(X, t) = t + (1 - \beta)^{-1}(X - t)_+, \tag{3}$$

with $\mathcal{T} = \mathbb{R}$, $\beta \in (0, 1)$ is the confidence level and $(x)_+ = \max\{x, 0\}$. CVaR_β is also known as expected shortfall. It is origin lies in financial mathematics,[9,11] but owing to Kouri[12] and Kouri and Surowiec,[6] it is now being widely used in engineering applications. Our work in particular, focuses on minimization problems (1) with $\mathcal{R}$ given by CVaR_β but it can be extended to other coherent risk measures, such as buffered probability of exceedence (BPOE), of type (2).

Notice that, since risk measures, such as CVaR_β focus on the upper tail events, the traditional sampling techniques to solve these stochastic PDE-constrained optimization problems are often computationally expensive. More precisely, CVaR_β captures the cost associated with rare events, but it requires more samples in order to be accurately approximated, which leads to many differential equation solves.[6] Moreover, the presence of the nonsmooth function $(\cdot)_+$ in CVaR_β poses several challenges, including, nondifferentiable cost functional, wasted Monte Carlo samples outside of the support of $(\cdot)_+ = \max\{\cdot, 0\}$, or slowly converging polynomial and other function approximation methods.

To tackle some of these challenges,[6] has proposed a smoothing of $(\cdot)_+$ which requires solving a sequence of smoothed optimization problems using Newton-based methods. Another solution strategy is to reformulate the problem and use interior-point methods.[8] A duality-based approach has also been recently proposed in Reference 13.

In this paper we develop an efficient method to tackle the above challenges associated with minimization of CVaR_β subject to constraints given by differential equations with random inputs. We consider two formulations of (1). The first one is the implicit approach where we remove the equality constraint $c(y, u; \omega) = 0$ via a control to solution map $u \mapsto y$.[6,8,13] The second case is the full space approach, where we directly tackle the full problem (1) using the Lagrangian formulation. The latter formulation appears to be new in the context of risk-averse optimization. Numerical experiments demonstrate that the full formulation converges more reliably for extreme parameters, for example, large β and small α.

Our framework builds rigorously on tensor decomposition methods, which emerged in the past two decades[14,15] as an efficient approximation of multi-index arrays, in particular when those contain expansion coefficients of high-dimensional functions.[16,17] The idea starts from the classical separation of variables. Functions of certain structure[18] or regularity[19] have been shown to admit rapidly (often exponentially) convergent series, where each term is a product of univariate functions. Later, instead of a simple sum of products, it was found more practical to consider hierarchical separation of variables.[14] A particularly simple instance of such is the Tensor Train (TT) decomposition[20] that admits efficient numerical computations. One of the most powerful algorithms of this kind is the cross approximation, as well as its variants.[21-23] Those allow one to compute a TT approximation to potentially any function, using a number of samples from the sought function that is a small multiple of the number of degrees of freedom in the tensor decomposition. Once TT decompositions are computed, integration, differentiation and linear algebra of the original functions can be implemented using their TT formats instead with a linear cost in the dimension.

However, irregular functions, such as the $(\cdot)_+$ function in (3), may lack an efficient TT decomposition. The main novelty of the paper is an algorithm that is adaptive in both the TT complexity and width parameter in the smoothed CVaR function, which allows one to actually alleviate the curse of dimensionality, since smooth functions do admit convergent TT approximation. If the bias from the smoothing is still too large for a feasible TT decomposition, we can obtain an unbiased, asymptotically exact solution with a version of Multilevel Monte Carlo methods,[24] namely, we use a smoothed solution as a control variate.[25]

The numerical experiments demonstrate that the stochastic risk-averse control problem can be solved with a cost that depends at most polynomially on the dimension. This allows us to solve a realistic risk-averse ODE control problem with 20 random variables.

1.1 | Outline

Section 2 and Appendix set up the relevant notation and provide the necessary background on risk-averse-optimization and TT decomposition. In Section 3, we introduce the control-to-state map, that is, $u \mapsto y$, and eliminate the equality constraints $c(\cdot) = 0$. The resulting optimization problem (1) is only a function of the control variable u and is known as reduced problem. A control variate correction of the problem is considered in Section 3.2. When the constraint $c(\cdot)$ is handled directly, the resulting formulation is called full-space problem and is discussed in Section 4. This is followed by Section 4.2 where the Gauss–Newton system for the full-space formulation is considered. Next, in Section 4.3 we consider preconditioning strategies for this formulation. Section 5 focuses on our numerical examples, where we first consider an optimal control problem constrained by an elliptic PDE with random coefficients. This is followed by the risk-averse optimal control of an infectious disease model which has been recently developed to propose lockdown strategies in the United Kingdom due to COVID-19.

2 | BACKGROUND

In this section, we first provide background on CVaR_β and introduce a regularized problem with CVaR_β replaced by $\text{CVaR}_\beta^\varepsilon$ with $\varepsilon > 0$. This is followed by a discussion on TT decomposition. The aim of this section is to set the stage for TTRISK.

2.1 | Risk-averse optimization

Let $\mathcal{L} := (\Omega, \mathcal{A}, \mathbb{P})$ be a complete probability space. Here, Ω is the set of outcomes, $\mathcal{A} \subset 2^\Omega$ is the σ-algebra of events, and $\mathbb{P} : \mathcal{A} \to [0, 1]$ is an appropriate probability measure. Let X be a (scalar) random variable defined on $\mathcal{L}$. For example, we will consider the objective function $\mathcal{J}$ in what follows. Then, the expectation of X denoted by $\mathbb{E}[X]$ is given by

$$\mathbb{E}[X] = \int_\Omega X(\omega) d\mathbb{P}(\omega).$$

As stated in the introduction, this paper focuses on optimization problems of type (1) where $\mathcal{R}$ is the risk measure given by (2). In particular, we consider the conditional value-at-risk (CVaR_β) at confidence level $\beta, \beta \in (0, 1)$ where f in (2) is

given by (3), with $\mathcal{T} = \mathbb{R}$. CVaR_β builds on the concept of value-at-risk (VaR) at level $\beta \in (0,1)$, which is the β-quantile of a given random variable.

More precisely, let X be a random variable and let $\beta \in (0,1)$ be fixed. Then, $\text{VaR}_\beta[X]$ is given by

$$\text{VaR}_\beta[X] := \inf_{t\in\mathbb{R}} \{t : \ \mathbb{P}[X \le t] \ge \beta\},$$

where $\mathbb{P}[X \le t]$ denotes the probability that the random variable X is less than or equal to t. VaR_β is unfortunately not coherent because it violates subadditivity / convexity. This is why CVaR_β is preferred as a risk measure.

Even though now we know that coherent risk-measures can be written in the abstract form (2), however, the one-dimensional (1D) minimization formulation of CVaR_β was first introduced by Rockafellar and Uryasev in References 9,11:

$$\text{CVaR}_\beta[X] = \inf_{t\in\mathbb{R}} \left\{ t + \frac{1}{1-\beta}\mathbb{E}[(X-t)_+] \right\}.$$

Moreover, if X is a continuous random variable, then

$$\text{CVaR}_\beta[X] = \mathbb{E}[X | X > \text{VaR}_\beta[X]],$$

which shows that $\text{CVaR}_\beta[X]$ is the average of β-tail of the distribution of X. Thus, $\text{CVaR}_\beta[X]$ emphasizes rare and low probability events, especially when $\beta \to 1$.

2.2 | Model problem

Our model problems are obtained by replacing $\mathcal{R}$ in (1) by CVaR_β, that is,

$$\min_{u\in U_{ad}} \text{CVaR}_\beta[\mathcal{J}(y,u;\omega)] + \alpha\mathcal{P}(u) \text{ subject to } c(y,u;\omega) = 0, \quad \text{in } Z, \quad \text{a.a. } \omega \in \Omega, \tag{4}$$

where again U_{ad} denotes the set of admissible controls. We use (y,u) to denote the state and control, respectively. In our setting u is always deterministic. The constraint equation $c(y,u;\omega) = 0$ represents a differential equation with uncertain coefficients, $\mathcal{P}(u)$ is a deterministic cost function, α is the regularization parameter, and $\mathcal{J}(y,u;\omega)$ is a random variable cost function.

Here, we make the *finite dimensional noise assumption* on the equality constraint.[6] We assume that ω can be sampled via a finite random vector $\xi : \Omega \to \Xi$ instead, where $\Xi := \xi(\Omega) \subset \mathbb{R}^d$ with $d \in \mathbb{N}$. For example, coefficients, defining the constraint $c(y,u;\omega) = 0$, may be expressed by a Karhunen–Loeve (KL) approximation of an infinite-dimensional continuous random field, see (42) for an example. This allows us to redefine the probability space to (Ξ, Σ, ρ), where $\Sigma = \xi(\mathcal{A})$ is the σ-algebra of regions, and $\rho(\xi)$ is the continuous probability density function such that $\mathbb{E}[X] = \int_\Xi X(\xi)\rho(\xi)d\xi$. The random variable $X(\xi)$ can be considered as a function of the random vector $\xi = (\xi^{(1)}, \dots, \xi^{(d)})$, belonging to the Hilbert space $\mathcal{F} = \{X(\xi) : \|X\| < \infty\}$, equipped with the inner product $\langle X, Y\rangle = \int_\Xi X(\xi)Y(\xi)\rho(\xi)d\xi$ and the Euclidean norm $\|X\| = \sqrt{\langle X, X\rangle}$. Since $\xi^{(1)}, \dots, \xi^{(d)}$ are independent random variables, we assume that each $\xi^{(k)}$ has a probability density function $\rho^{(k)}(\xi^{(k)})$, and that the space of functions $\mathcal{F}$ is isomorphic to a tensor product of spaces of univariate functions, $\mathcal{F} = \mathcal{F}^{(1)} \otimes \cdots \otimes \mathcal{F}^{(d)}$, where $\mathcal{F}^{(k)} = \{X^{(k)}(\xi^{(k)}) : \|X^{(k)}\| < \infty\}$, $\|X^{(k)}\| = \sqrt{\langle X^{(k)}, X^{(k)}\rangle}$, $\langle X^{(k)}, Y^{(k)}\rangle = \int_{\mathbb{R}} X^{(k)}(\xi^{(k)})Y^{(k)}(\xi^{(k)})\rho^{(k)}(\xi^{(k)})d\xi^{(k)}$, $k = 1, \dots, d$.

Then, (4) reads

$$\min_{u\in U_{ad}} \text{CVaR}_\beta[\mathcal{J}(y,u;\xi)] + \alpha\mathcal{P}(u) \text{ subject to } c(y,u;\xi) = 0, \quad \text{in } Z, \quad \text{a.a. } \xi \in \Xi. \tag{5}$$

Here and in what follows, bracketed superscripts (e.g. $\xi^{(d)}$) denote a component, not a power or derivative.

To tackle nonsmoothness in CVaR_β, we employ a smoothing-based approach from.[6] The smoothing approach is essentially aimed at approximating the positive part function $(\cdot)_+$ in CVaR_β by a smooth function $g_\varepsilon : \mathbb{R} \to \mathbb{R}$, which depends on some $\varepsilon > 0$. Various examples of g_ε are available in Reference 6 (section 4.1.1). In particular, we consider the following

C^∞-smoothing function

$$g_\varepsilon(x) = \varepsilon \log(1 + \exp(x/\varepsilon)), \tag{6}$$

where

$$g'_\varepsilon(x) = \frac{1}{1+\exp(-x/\varepsilon)}, \qquad g''_\varepsilon(x) = \frac{1}{\varepsilon}\left(\frac{1}{\exp(x/(2\varepsilon)) + \exp(-x/(2\varepsilon))}\right)^2. \tag{7}$$

Thus, the optimization problem for smooth $\text{CVaR}_\beta^\varepsilon$ is given by

$$\begin{cases} \min_{(u,t)\in U_{ad}\times\mathbb{R}} \ \mathcal{R}^\varepsilon_{t,\beta}[\mathcal{J}(y,u;\xi)] + \alpha\mathcal{P}(u) \\ \text{subject to} \\ c(y,u;\xi) = 0, \quad \text{in } Z, \quad \text{a.a. } \xi \in \Xi, \end{cases}, \tag{8}$$

where

$$\mathcal{R}^\varepsilon_{t,\beta}[\mathcal{J}(y,u;\xi)] := t + \frac{1}{1-\beta}\mathbb{E}[g_\varepsilon(\mathcal{J}(y,u;\xi) - t)]. \tag{9}$$

For convergence analysis of (8) to (5), we refer to Reference 6.

2.3 | Cartesian function space

The dimension d of the random vector ξ can be arbitrarily high, for example, tens of model tuning parameters or KL coefficients. In this case expectations as in (9) become high-dimensional integrals. Instead of a direct Monte Carlo average (which may converge too slowly), we can introduce a high-order quadrature rule (e.g. Gauss–Legendre) with $n_\xi \in \mathbb{N}$ points in each of the components $\xi^{(1)}, \dots, \xi^{(d)}$ independently. However, the exponential total number of quadrature points in all variables n_ξ^d becomes intractable even for moderate dimensions.

2.4 | TT decomposition

We circumvent this "curse of dimensionality" problem by approximating all functions depending on ξ by a TT decomposition,[20] which admits efficient integration and differentiation.

Definition 1. A square-integrable function $f(\xi)$ is said to be approximated by a (functional) TT decomposition[16,17] with a relative approximation error ϵ if there exist univariate functions $F^{(k)}(\cdot) : \xi^{(k)} \in \mathbb{R} \to \mathbb{R}^{r_{k-1}\times r_k}$, $k = 1, \dots, d$, such that

$$\tilde{f}(\xi) := \sum_{s_0,\dots,s_d=1}^{r_0,\dots,r_d} F^{(1)}_{s_0,s_1}(\xi^{(1)}) F^{(2)}_{s_1,s_2}(\xi^{(2)}) \cdots F^{(d)}_{s_{d-1},s_d}(\xi^{(d)}), \tag{10}$$

where the subscripts s_{k-1}, s_k denote elements of a matrix, and $\|f - \tilde{f}\| = \epsilon\|f\|$. The factors $F^{(k)}$ are called TT cores, and the ranges of summation indices $r_0, \dots, r_d \in \mathbb{N}$ are called TT ranks.

Without loss of generality we can let $r_0 = r_d = 1$, but the other TT ranks $r_1, \dots, r_{d-1}$ can vary depending on the approximation error. One example is a bi-variate truncated Fourier series $\tilde{f}(\xi^{(1)}, \xi^{(2)}) = \sum_{s=-r}^{r} f_s(\xi^{(1)}) \exp(\mathrm{i}s\xi^{(2)})$.

From (10), we notice that the expectation of $\tilde{f}$ factorizes into univariate integrations,

$$\mathbb{E}[\tilde{f}] = \sum_{s_0,\dots,s_d=1}^{r_0,\dots,r_d} \left(\int F^{(1)}_{s_0,s_1}(\xi^{(1)})\rho^{(1)}(\xi^{(1)})d\xi^{(1)}\right) \cdots \left(\int F^{(d)}_{s_{d-1},s_d}(\xi^{(d)})\rho^{(d)}(\xi^{(d)})d\xi^{(d)}\right).$$

For practical computations with (10) we introduce univariate bases $\{\ell_i(\xi^{(k)})\}_{i=1}^{n_\xi}$, and the multivariate basis constructed from a tensor product,

$$L_{i_1,\dots,i_d}(\xi) := \ell_{i_1}(\xi^{(1)})\cdots\ell_{i_d}(\xi^{(d)}).$$

Now we can collect the expansion coefficients of $\tilde{f}$ into a tensor $\mathbf{F}\in\mathbb{R}^{n_\xi\times\cdots\times n_\xi}$,

$$\tilde{f}(\xi) = \sum_{i_1,\dots,i_d=1}^{n_\xi} \mathbf{F}(i_1,\dots,i_d)L_{i_1,\dots,i_d}(\xi). \tag{11}$$

Similarly, TT cores in (10) can be written using three-dimensional tensors $\mathbf{F}^{(k)}\in\mathbb{R}^{r_{k-1}\times n_\xi\times r_k}$,

$$F^{(k)}_{s_{k-1},s_k}(\xi^{(k)}) = \sum_{i=1}^{n_\xi}\mathbf{F}^{(k)}(s_{k-1},i,s_k)\ell_i(\xi^{(k)}),\quad k=1,\dots,d. \tag{12}$$

The original (discrete) TT decomposition[20] was introduced for tensors,

$$\mathbf{F}(i_1,\dots,i_d) = \sum_{s_0,\dots,s_d=1}^{r_0,\dots,r_d}\mathbf{F}^{(1)}(s_0,i_1,s_1)\cdots\mathbf{F}^{(d)}(s_{d-1},i_d,s_d). \tag{13}$$

Note that $\mathbf{F}$ contains n_ξ^d elements, whereas storing $\mathbf{F}^{(1)},\dots,\mathbf{F}^{(d)}$ needs only $\sum_k r_{k-1}n_\xi r_k$ elements. For brevity we can define the maximal TT rank $r := \max_k r_k$, which gives us a linear storage complexity of the TT decomposition, $\mathcal{O}(dn_\xi r^2)$.

If $\{\ell_i(\xi^{(k)})\}_{i=1}^{n_\xi}$ is a Lagrange polynomial basis, defined by the quadrature points $\Xi^{(k)} := \{\xi_i^{(k)}\}$ and weights $\{w_i\}$ such that $\ell_i(\xi_j^{(k)}) = \delta_{i,j}$, we obtain

$$\begin{aligned}\mathbb{E}[\tilde{f}] &= \sum_{i_1,\dots,i_d}\tilde{f}(\xi_{i_1}^{(1)},\dots,\xi_{i_d}^{(d)})w_{i_1}\cdots w_{i_d} = \sum_{i_1,\dots,i_d}\mathbf{F}(i_1,\dots,i_d)w_{i_1}\cdots w_{i_d}\\ &= \sum_{s_0,\dots,s_d=1}^{r_0,\dots,r_d}\left(\sum_{i_1}\mathbf{F}^{(1)}(s_0,i_1,s_1)w_{i_1}\right)\cdots\left(\sum_{i_d}\mathbf{F}^{(d)}(s_{d-1},i_d,s_d)w_{i_d}\right).\end{aligned} \tag{14}$$

The summation over $s_0,\dots,s_d$ in the right hand side can be computed recursively by multiplying only two tensors at a time. Assuming that a partial result $\mathbf{R}_k\in\mathbb{R}^{r_0\times r_k}$ is given, we can compute

$$\mathbf{R}_{k+1}(s_0,s_{k+1}) = \sum_{s_k=1}^{r_k}\mathbf{R}_k(s_0,s_k)\left(\sum_{i_{k+1}}\mathbf{F}^{(k+1)}(s_k,i_{k+1},s_{k+1})w_{i_{k+1}}\right), \tag{15}$$

as a matrix product with a $\mathcal{O}(n_\xi r^2)$ complexity. Starting with $\mathbf{R}_0 = 1$ and finishing with $\mathbf{R}_d = \mathbb{E}[\tilde{f}]$, we complete the entire integration in $\mathcal{O}(dn_\xi r^2)$ operations.

Such a recursive sweep over TT cores is paramount to computing TT approximations of arbitrary functions, or to solution of operator equations in the TT format. For example, the *TT-Cross* method (see Appendix A) requires $\mathcal{O}(dn_\xi r^2)$ samples from a function $f(\xi)$ and $\mathcal{O}(dn_\xi r^3)$ further floating point operations to compute a TT approximation $\tilde{f}(\xi)\approx f(\xi)$. Similarly, linear algebra on functions can be recast to linear algebra on their TT cores with a linear complexity in the dimension (see Appendix B).

3 | REDUCED SPACE FORMULATION

This paper considers two approaches to tackle (8). In this section, we first introduce TTRISK for the so-called reduced form of (8). To ensure that $\mathrm{CVaR}_\beta^\varepsilon$ is a statistically unbiased estimator of CVaR_β, we introduce a control variate correction in Section 3.2.

3.1 | Smoothed CVaR with TT approximations

Assume that $c(y,u;\xi)=0$ is uniquely solvable, that is, for each $u\in U_{ad}$ there exists a unique solution mapping $y(u;\cdot):\Xi\to Y$ for $\mathbb{P}$ a.a $\xi\in\Xi$. The resulting optimization problem (8) only depends on u and is given by

$$\min_{(u,t)\in U_{ad}\times\mathbb{R}}\left\{\mathfrak{J}(u,t):=\mathcal{R}^{\epsilon}_{t,\beta}[j(u;\xi)]+\alpha\mathcal{P}(u)\right\}, \tag{16}$$

where

$$j(u;\xi):=\mathcal{J}(y(u;\xi),u;\xi). \tag{17}$$

The exact expectation in $\mathrm{CVaR}^{\epsilon}_{\beta}$ can be approximated by a quadrature similarly to (14). We denote the total number of quadrature points N (which is formally $N=n_{\xi}^{d}$ in the Cartesian formulation). However, we need to tackle the curse of dimensionality using the TT decomposition.

Definition 2. The approximate expectation $\mathbb{E}_N[f]$ of a function $f(\xi)$ is defined as $\mathbb{E}_N[f]:=\mathbb{E}[\tilde{f}]$, where $\tilde{f}(\xi)$ is a TT approximation (10) to $f(\xi)$, computed using the TT-Cross algorithm as described in Appendix A, and the integration of $\tilde{f}(\xi)$ is carried out as shown in (15).

This leads to the following approximation of (16):

$$\min_{(u,t)\in U_{ad}\times\mathbb{R}}\left\{\mathfrak{J}_N(u,t):=\mathcal{R}^{\epsilon}_{t,\beta,N}[j(u;\xi)]+\alpha\mathcal{P}(u)\right\}, \tag{18}$$

where

$$\mathcal{R}^{\epsilon}_{t,\beta,N}[j(u;\xi)]:=t+\frac{1}{(1-\beta)}\mathbb{E}_N[g_{\epsilon}(j(u;\xi)-t)].$$

However, to optimize the entire cost $\mathfrak{J}_N$ we need to calculate the first and second-order derivatives. We readily obtain that the first-order derivatives are

$$\begin{aligned}\nabla_u\mathfrak{J}_N(u,t)&=(1-\beta)^{-1}\mathbb{E}_N[g'_{\epsilon}(j(u;\xi)-t)\nabla_u j(u;\xi)]+\alpha\nabla_u\mathcal{P}(u)\\ \nabla_t\mathfrak{J}_N(u,t)&=1-(1-\beta)^{-1}\mathbb{E}_N[g'_{\epsilon}(j(u;\xi)-t)],\end{aligned} \tag{19}$$

and the second-order derivatives are

$$\begin{aligned}\nabla_{uu}\mathfrak{J}_N(u,t)=(1-\beta)^{-1}\mathbb{E}_N&\left[g''_{\epsilon}(j(u;\xi)-t)\nabla_u j(u;\xi)\nabla_u j(u;\xi)^{*}+g'_{\epsilon}(j(u;\xi)-t)\nabla_{uu}j(u;\xi)\right]\\&+\alpha\nabla_{uu}\mathcal{P}(u),\end{aligned} \tag{20}$$

$$\nabla_{ut}\mathfrak{J}_N(u,t)=-(1-\beta)^{-1}\mathbb{E}_N[g''_{\epsilon}(j(u;\xi)-t)\nabla_u j(u;\xi)], \tag{21}$$

$$\nabla_{tu}\mathfrak{J}_N(u,t)=-(1-\beta)^{-1}\mathbb{E}_N[g''_{\epsilon}(j(u;\xi)-t)\nabla_u j(u;\xi)^{*}], \tag{22}$$

$$\nabla_{tt}\mathfrak{J}_N(u,t)=(1-\beta)^{-1}\mathbb{E}_N[g''_{\epsilon}(j(u;\xi)-t)]. \tag{23}$$

Observe that the second derivatives of $\mathfrak{J}_N$ computed above depend on g''_{ϵ}. From (7), we see that g''_{ϵ} decays rapidly away from the interval $(-\epsilon,\epsilon)$. Consequently, the Hessian

$$H=\begin{bmatrix}\nabla_{uu}\mathfrak{J}_N & \nabla_{ut}\mathfrak{J}_N\\ \nabla_{tu}\mathfrak{J}_N & \nabla_{tt}\mathfrak{J}_N\end{bmatrix}, \tag{24}$$

can degenerate if $g''_{\epsilon}(j(u;\xi_i)-t)=0$ for all $i=1,\ldots,N$, since all but the $(1,1)$ block of H are zeros.[26] (In fact, the Hessian may become ill-conditioned also if $g''_{\epsilon}(j(u;\xi_i)-t)$ is close to zero.) To circumvent this problem, we adopt a technique

similar to that used in augmented Lagrangian methods to dynamically update augmentation parameters within the optimization problem.[26,27] We start with $t_0 = \varepsilon_0 = \mathbb{E}_N[j(u_0;\xi)]$. In many cases there will be a lot of points $j(u;\xi_i) - t$ in a significant support of g''_ε. During the course of Newton iterations, we decrease ε geometrically,

$$\varepsilon_{m+1} = \mu_\varepsilon \varepsilon_m, \tag{25}$$

where $0 < \mu_\varepsilon < 1$ is a tuning factor (e.g. $\mu_\varepsilon = 1/2$). However, the next iterate t_{m+1} may end up far on the tail of g''_{ε_m} again. To prevent this from happening, we perform a line search, where in addition to a non-increasing residual condition[1]

$$\left\| \begin{bmatrix} \nabla_u \mathfrak{J}_N(u_{m+1}, t_{m+1}) \\ \nabla_t \mathfrak{J}_N(u_{m+1}, t_{m+1}) \end{bmatrix} \right\| \le \left\| \begin{bmatrix} \nabla_u \mathfrak{J}_N(u_m, t_m) \\ \nabla_t \mathfrak{J}_N(u_m, t_m) \end{bmatrix} \right\|, \tag{26}$$

where $u_{m+1} = u_m + h\delta_{um}$, $t_{m+1} = t_m + h\delta_{tm}$ with a step size $h > 0$ and Newton directions δ_{um}, δ_{tm}, we require that

$$\mathbb{E}_N[\exp(-|j(u_{m+1};\xi) - t_{m+1}|/\varepsilon_m)] > \theta, \tag{27}$$

for some $0 < \theta < 1$. In the numerical experiments we have found the iterations to be robust and insensitive for θ between 10^{-2} and 10^{-1}.

Proposition 1. *This ensures that*

$$\nabla_{tt} \mathfrak{J}_N(u_{m+1}, t_{m+1}) > \frac{\theta^{1/\mu_\varepsilon}}{4\varepsilon_{m+1}(1-\beta)},$$

stays away from zero by a fixed fraction of the maximum of $g''_{\varepsilon_{m+1}}(t)$, *which is equal to* $1/(4\varepsilon_{m+1})$.

Proof. Firstly, note from (7) that

$$g''_\varepsilon(t) \ge \frac{1}{\varepsilon}\left(\frac{1}{2\exp(|t|/(2\varepsilon))}\right)^2 = \frac{1}{4\varepsilon}\exp\left(-\frac{|t|}{\varepsilon}\right). \tag{28}$$

Now,

$$\begin{aligned}
\nabla_{tt}\mathfrak{J}_N(u_{m+1}, t_{m+1}) &= (1-\beta)^{-1}\mathbb{E}_N\left[g''_{\varepsilon_{m+1}}(j(u_{m+1};\xi) - t_{m+1})\right] && \text{(from Equation 23)} \\
&\ge \frac{1}{4\varepsilon_{m+1}(1-\beta)}\mathbb{E}_N\left[\exp\left(-\frac{|j(u_{m+1};\xi) - t_{m+1}|}{\mu_\varepsilon \varepsilon_m}\right)\right] && \text{(by linearity of } \mathbb{E}_N, (28) \text{ and } (25)) \\
&= \frac{1}{4\varepsilon_{m+1}(1-\beta)}\mathbb{E}_N\left[\exp\left(-\frac{|j(u_{m+1};\xi) - t_{m+1}|}{\varepsilon_m}\right)^{1/\mu_\varepsilon}\right] \\
&\ge \frac{1}{4\varepsilon_{m+1}(1-\beta)}\left(\mathbb{E}_N\left[\exp\left(-\frac{|j(u_{m+1};\xi) - t_{m+1}|}{\varepsilon_m}\right)\right]\right)^{1/\mu_\varepsilon} && \text{(by Jensen's inequality since } \frac{1}{\mu_\varepsilon} > 1) \\
&> \frac{1}{4\varepsilon_{m+1}(1-\beta)}\theta^{1/\mu_\varepsilon}. && \text{(by assumption (27))}
\end{aligned}$$

The proof is complete. ∎

If the dimension of discretized u is small, we can compute the TT-Cross approximation of $\nabla_u j(u_m, \xi)$, $\nabla_{uu} j(u_m, \xi)$, $g''_{\varepsilon_m}(j(u_m;\xi) - t_m)$ and $g''_{\varepsilon_m}(j(u_m;\xi) - t_m)$ directly, since evaluations of j, $\nabla_u j$ and $\nabla_{uu} j$ are available explicitly from (17) under the unique solution mapping $y(u;\cdot)$, compute the expectations in (19) and (20), and solve

$$H\begin{bmatrix}\delta_{um} \\ \delta_{tm}\end{bmatrix} = -\begin{bmatrix} \nabla_u \mathfrak{J}_N(u_m, t_m) \\ \nabla_t \mathfrak{J}_N(u_m, t_m)\end{bmatrix}, \tag{28}$$

for the Newton directions. However, if u is large, $\nabla_{uu} j(u_m, \xi)$ is large and dense, and its TT decomposition becomes too expensive. In fact, even multiplying $\nabla_{uu}\mathfrak{J}_N(u,t)$ by a vector in an iterative solver would require recomputation of the TT

decomposition in each iteration. To avoid this problem, we propose to replace the expectation in (20) by sampling at a fixed point $\bar{\xi}$. This gives

$$\nabla_{uu}\widetilde{\mathfrak{J}_N}(u,t) = (1-\beta)^{-1}\left[\mathbb{E}_N[g''_\varepsilon(j(u;\xi)-t)]\nabla_u j(u;\bar{\xi})\nabla_u j(u;\bar{\xi})^*\right] + (1-\beta)^{-1}\mathbb{E}_N[g'_\varepsilon(j(u;\xi)-t)]\nabla_{uu}j(u;\bar{\xi}) + \alpha\nabla_{uu}\mathcal{P}(u), \tag{29}$$

and consequently a *fixed-point Hessian*

$$\tilde{H} = \begin{bmatrix} \nabla_{uu}\widetilde{\mathfrak{J}_N} & \nabla_{ut}\mathfrak{J}_N \\ \nabla_{tu}\mathfrak{J}_N & \nabla_{tt}\mathfrak{J}_N \end{bmatrix}. \tag{30}$$

The choice of $\bar{\xi}$ is motivated by the mean value theorem. We can treat $\mathbb{E}_N[g'_\varepsilon(j(u;\xi)-t)\nabla_{uu}j(u;\xi)]$ as an expectation of $\nabla_{uu}j(u;\xi)$ alone over a probability density function $\bar{\rho}(\xi) = (1/C)g'_\varepsilon(j(u;\xi)-t)\rho(\xi)$, where $C := \mathbb{E}_N[g'_\varepsilon(j(u;\xi)-t)]$ is the normalizing constant. In turn, for a linear $\nabla_{uu}j(u;\xi)$ it holds $\mathbb{E}_{\bar{\rho}}[\nabla_{uu}j(u;\xi)] = \nabla_{uu}j(u;\mathbb{E}_{\bar{\rho}}[\xi])$, so we can take the right-hand side as an approximation also in a general case. This gives

$$\bar{\xi} = \mathbb{E}_{\bar{\rho}}[\xi] = \frac{\mathbb{E}_N[g'_\varepsilon(j(u;\xi)-t)\xi]}{\mathbb{E}_N[g'_\varepsilon(j(u;\xi)-t)]}. \tag{31}$$

We focus on $\nabla_{uu}j$ (and hence on g'_ε) since $\nabla_{uu}j$ is usually the dominant part of $\nabla_{uu}\mathfrak{J}_N$. Note that the action $\nabla_{uu}j(u;\bar{\xi})\cdot\delta_u$ can usually be applied efficiently, since this requires the solution of one forward and one adjoint deterministic problems at fixed $\xi = \bar{\xi}$. Similarly, $\nabla_{uu}\mathcal{P}(u)$ is a sparse, and $\nabla_u j(u;\bar{\xi})\nabla_u j(u;\bar{\xi})^*$ is a rank-1 matrix after discretization. This allows us to solve the Newton system (28) with $\tilde{H}$ instead of H iteratively with fast matrix-vector products.

Lastly, if U_{ad} is constrained, we can add the projection of the Newton direction onto U_{ad}. To reduce the number of projections in the step selection stage, we write the method in a Frank–Wolfe's fashion,[28] that is, we project the search direction, $\hat{\delta}_{um} := \mathrm{Proj}_{U_{ad}}(u_m+\delta_{um}) - u_m$, where $\mathrm{Proj}_{U_{ad}}(\cdot)$ is the orthogonal projection onto U_{ad}, followed by the usual line search in $u_{m+1} = u_m + h\hat{\delta}_{um}$. The entire procedure is summarized in Algorithm 1.

3.2 | Smoothed CVaR as control variate for Monte Carlo

Assuming that $\tilde{g}_\varepsilon$ and $\tilde{j}$ are TT approximations to g_ε and j, respectively, we can define the approximate expectation of the exact function as the exact expectation of the approximate function, since the approximate function is a polynomial:

$$\mathbb{E}_N[g_\varepsilon(j(u;\xi)-t)] = \mathbb{E}[\tilde{g}_\varepsilon(\tilde{j}(u;\xi)-t)].$$

Algorithm 1. TTRISK (REDUCED CASE)

Require: α, β, line search parameter θ, number of iterations $I_{\max}$, smoothness reduction factor μ_ε, stopping tolerance, procedure to compute $j(u,\xi)$.

1: Set $m=0$, $u_0=0$, $t_0=\varepsilon_0=\mathbb{E}_N[j(u_0;\xi)]$.
2: **while** $m \le I_{\max}$ and $|t_m - t_{m-1}| > \mathrm{tol}\cdot|t_m|$ or $\|u_m-u_{m-1}\| > \mathrm{tol}\cdot\|u_m\|$ or $m=0$ **do**
3: Approximate $j(u_m;\xi)$, $\nabla_u j(u_m;\xi)$, $g'_{\varepsilon_m}(j(u_m;\xi)-t_m)$, $g''_{\varepsilon_m}(j(u_m;\xi)-t_m)$ by TT-Cross.
4: Compute expectations in (19), (21)–(23), (29) and (31) using (14),(15).
5: Solve (28) with H or $\tilde{H}$ using Conjugate Gradients method.
6: Project the increment $\hat{\delta}_{um} := \mathrm{Proj}_{U_{ad}}(u_m+\delta_{um}) - u_m$. ▷ for control constraints
7: Find $0<h\le 1$ such that (26) and (27) hold.
8: Update $u_{m+1} = u_m + h\hat{\delta}_{um}$, $t_{m+1} = t_m + h\delta_{tm}$, $\varepsilon_{m+1} = \mu_\varepsilon\varepsilon_m$.
9: Set $m = m+1$
10: **end while**
11: **return** $u_m \approx u$, $t_m \approx t$, $\mathcal{R}^{\varepsilon_m}_{t,\beta,N}[j(u_m;\xi)] \approx \mathrm{CVaR}_\beta$.

Using add-and-subtract trick, we can write the exact risk-measure functional as follows,

$$\mathcal{R}_t = t + \frac{1}{1-\beta}\mathbb{E}[\tilde{g}_\epsilon(\tilde{j}(u;\xi)-t)] + \frac{1}{1-\beta}\left[\mathbb{E}(j(u;\xi)-t)_+ - \mathbb{E}[\tilde{g}_\epsilon(\tilde{j}(u;\xi)-t)]\right].$$

However, for the last term we can use a different quadrature for computing the expectation, such as Monte Carlo with M samples. This defines a *corrected* smoothed functional:

$$\begin{aligned}\mathcal{R}^{\epsilon,M}_{t,\beta,N}[j(u;\xi)] &:= t + \frac{1}{(1-\beta)}\mathbb{E}_N[\tilde{g}_\epsilon(\tilde{j}(u;\xi)-t)] \\ &\quad + \frac{1}{(1-\beta)}\frac{1}{M}\sum_{\ell=1}^{M}\left[(j(u;\xi_\ell)-t)_+ - \tilde{g}_\epsilon(\tilde{j}(u;\xi_\ell)-t)\right], \end{aligned} \tag{32}$$

where ξ_ℓ are i.i.d. samples from $\rho(\xi)$. The benefit of this scheme stems from the fact that if the approximation is accurate, that is, $\mathrm{var}[(j(u;\xi)-t)_+ - \tilde{g}_\epsilon(\tilde{j}(u;\xi)-t)] \le \delta^2\mathrm{var}[(j(u;\xi)-t)_+]$ is small (where δ is the total error from the TT-Cross (Algorithm 2) and smoothing), by the law of large numbers the variance of this estimator reads

$$\mathrm{var}\left[\mathcal{R}^{\epsilon,M}_{t,\beta,N}\right] \le \delta^2\frac{\mathrm{var}[(j(u;\xi)-t)_+]}{M} \ll \frac{\mathrm{var}[(j(u;\xi)-t)_+]}{M},$$

where the latter term is the variance of the straightforward Monte Carlo approximation of $\mathcal{R}_t$. Consequently, one needs a much smaller M to achieve the same variance (i.e. Mean Square Error) threshold. One can say that $\tilde{g}_\epsilon$ is used as a *control variate* for variance reduction of Monte Carlo,[25,29] or vice versa, that (32) is the *second-level correction*[24,30] to the surrogate model $\tilde{g}_\epsilon$. This makes $\mathcal{R}^{\epsilon,M}_{t,\beta,N}$ a statistically unbiased estimator of $\mathcal{R}_t$.

Similarly we can correct the cost function and its gradient (19):

$$\begin{aligned}\nabla_u\mathfrak{F}^M_N(u,t) &= (1-\beta)^{-1}\mathbb{E}[\tilde{g}'_\epsilon(\tilde{j}(u;\xi)-t)\nabla_u\tilde{j}(u;\xi)] + \alpha\nabla_u\mathcal{P}(u) \\ &\quad + \frac{1}{(1-\beta)M}\sum_{\ell=1}^{M}\left[\theta(j(u;\xi_\ell)-t)\nabla_u j(u;\xi_\ell) - \tilde{g}'_\epsilon(\tilde{j}(u;\xi_\ell)-t)\nabla_u\tilde{j}(u;\xi_\ell)\right], \end{aligned} \tag{33}$$

$$\begin{aligned}\nabla_t\mathfrak{F}^M_N(u,t) &= 1-(1-\beta)^{-1}\mathbb{E}[\tilde{g}'_\epsilon(\tilde{j}(u;\xi)-t)] \\ &\quad - \frac{1}{(1-\beta)M}\sum_{\ell=1}^{M}\left[\theta(j(u;\xi_\ell)-t) - \tilde{g}'_\epsilon(\tilde{j}(u;\xi_\ell)-t)\right], \end{aligned} \tag{34}$$

where $\theta(t) = 1$ if $t \ge 0$, and 0, otherwise. These gradients can be plugged into line 4 of Algorithm 1 instead of (19). Since the variance of the correction is expected to be small, we omit it in the Hessian, turning Algorithm 1 into a Gauss–Newton method.

4 | LAGRANGIAN CVAR FORMULATION

In this section, we first focus on the full-space formulation. This is followed by a Gauss–Newton system setup for the problem and a preconditioning strategy for this system.

4.1 | Semi-discrete formulation

As in Section 3, we let $\{\xi_i\}_{i=1}^N$ be all nodes, and $\{w_i\}_{i=1}^N$ be all weights in the quadrature (14). Moreover, we assume a Lagrangian basis expansion (11), and for brevity we let $L_i(\xi) := L_{i_1,\dots,i_d}(\xi)$ and $f_i := \tilde{f}(\xi_i) = \mathbf{F}(i_1,\dots,i_d)$. Applying this formalism to y instead of f, and assuming that the constraint $c(y,u,\xi) = 0$ holds pointwise in ξ, we obtain semi-discrete equations

$$c(y_i,u,\xi_i) = 0, \qquad i = 1,\dots,N.$$

Likewise, we can introduce the smoothed CVaR with the Monte Carlo correction (cf. (32)) using the quadrature

$$\mathcal{R}^{\epsilon,M}_{t,\beta,N} = t + (1-\beta)^{-1}\sum_{i=1}^{N} w_i g_\epsilon(\mathcal{J}(y_i,u,\xi_i)-t) + \frac{1}{(1-\beta)M}\sum_{\ell=1}^{M}\left[(\mathcal{J}(\tilde{y}(\xi_\ell),u,\xi_\ell)-t)_+ - g_\epsilon(\mathcal{J}(\tilde{y}(\xi_\ell),u,\xi_\ell)-t)\right].$$

Let $\vec{y}\in Y^N$ and $\vec{p}\in (Y^*)^N$ be functions y_i and p_i stacked together. We introduce the Lagrangian

$$\mathcal{L}(\vec{y},u,\vec{p},t) = \mathcal{R}^{\epsilon,M}_{t,\beta,N} + \alpha\mathcal{P}(u) + \sum_{i=1}^{N} w_i\langle p_i, c(y_i,u,\xi_i)\rangle.$$

In the differentiation, we will distinguish the components corresponding to different coefficients. This gives, for all $j = 1,\ldots,N$,

$$\begin{aligned}
\nabla_{y_j}\mathcal{L} &= \frac{1}{1-\beta}\left[w_j g_\epsilon'(\mathcal{J}(y_j,u,\xi_j)-t)\nabla_y\mathcal{J}(y_j,u,\xi_j) + \frac{1}{M}\sum_{\ell=1}^{M}E_j'(\xi_\ell)\right] + w_j(\nabla_y c)^* p_j,\\
\nabla_u\mathcal{L} &= \alpha\nabla_u\mathcal{P}(u) + \sum_{i=1}^{N} w_i(\nabla_u c(y_i,u,\xi_i))^* p_i,\\
\nabla_{p_j}\mathcal{L} &= w_j c(y_j,u,\xi_j),\\
\nabla_t\mathcal{L} &= 1-(1-\beta)^{-1}\left[\sum_{i=1}^{N} w_i g_\epsilon'(\mathcal{J}(y_i,u,\xi_i)-t) + \frac{1}{M}\sum_{\ell=1}^{M}e'(\xi_\ell)\right],
\end{aligned} \tag{35}$$

where we have defined error correction shortcuts

$$\begin{aligned}
e'(\xi) &= \left[\theta\left(\mathcal{J}\left(\tilde{y}(\xi),u,\xi\right)-t\right) - g_\epsilon'\left(\mathcal{J}\left(\tilde{y}(\xi),u,\xi\right)-t\right)\right],\\
E_j'(\xi) &= e'(\xi)L_j(\xi)\nabla_y\mathcal{J}\left(\tilde{y}(\xi),u,\xi\right),
\end{aligned} \tag{36}$$

with $\theta(t) = 1$ for $t\geq 0$ and 0 otherwise.

The second derivatives of (36) are difficult both notationally and computationally, since arbitrary points ξ_ℓ, leading to nonzero Lagrangian polynomial values $L_j(\xi_\ell)$, produce dense matrices. However, if we assume that the correction is small in magnitude, we can follow the arguments of Gauss–Newton methods and remove the correction derivatives in the Hessian, as well as second derivatives of $c(y,u,\xi)$ and $\mathcal{J}(y,u,\xi)$. This way we obtain (where $\delta_{j,k}$ denotes the Kronecker delta, and $[f(y,u,\xi)]_j$ evaluates $f(y_j,u,\xi_j)$)

$$\begin{aligned}
\nabla_{y_j,y_k}\mathcal{L} &\approx (1-\beta)^{-1}\delta_{j,k}w_j\left[g_\epsilon''\nabla_y\mathcal{J}(\nabla_y\mathcal{J})^* + g_\epsilon'\nabla_{yy}\mathcal{J}\right]_j, && j,k = 1,\ldots,N\\
\nabla_{y_j,p_k}\mathcal{L} &\approx \delta_{j,k}w_j(\nabla_y c(y_j,u,\xi_j))^*, && j,k = 1,\ldots,N\\
\nabla_{y_j,t}\mathcal{L} &\approx -(1-\beta)^{-1}w_j g_\epsilon''(\mathcal{J}(y_j,u,\xi_j)-t)\nabla_y\mathcal{J}(y_j,u,\xi_j), && j = 1,\ldots,N\\
\nabla_{u,p_k}\mathcal{L} &\approx (\nabla_u c(y_k,u,\xi_k))^* w_k, && k = 1,\ldots,N\\
\nabla_{p_j,y_k}\mathcal{L} &\approx \delta_{j,k}w_j\nabla_y c(y_j,u,\xi_j), && j,k = 1,\ldots,N\\
\nabla_{p_j,u}\mathcal{L} &\approx w_j\nabla_u c(y_j,u,\xi_j), && j = 1,\ldots,N\\
\nabla_{t,t}\mathcal{L} &\approx (1-\beta)^{-1}\sum_{i=1}^{N} w_i g_\epsilon''(\mathcal{J}(y_i,u,\xi_i)-t),\\
\nabla_{u,u}\mathcal{L} &\approx \alpha\nabla_{uu}\mathcal{P}(u),\\
\nabla_{p_j,p_k}\mathcal{L} &= 0,
\end{aligned}$$

as well as symmetric terms. For both notational and computational brevity, let us introduce

$$H_j^{yt} = -(1-\beta)^{-1} g_\varepsilon''(\mathcal{J}(y_j,u,\xi_j)-t)\nabla_y \mathcal{J}(y_j,u,\xi_j) \in Y$$
$$H_j^{yy} = (1-\beta)^{-1}\left[g_\varepsilon''(\mathcal{J}-t)\nabla_y\mathcal{J}(\nabla_y\mathcal{J})^* + g_\varepsilon'(\mathcal{J}-t)\nabla_{yy}\mathcal{J}\right]_j \in \mathcal{L}(Y,Y)$$
$$H^{tt} = (1-\beta)^{-1}\sum_{i=1}^N w_i g_\varepsilon''(\mathcal{J}(y_i,u,\xi_i)-t) \in \mathbb{R}_+.$$

The entire Newton KKT system can thus be written as follows,

$$\begin{bmatrix} \mathrm{bdiag}(w_j H_j^{yy}) & 0 & \mathrm{bdiag}(w_j(\nabla_y c(y_j,u,\xi_j))^*) & w\odot H^{yt} \\ 0 & \alpha\nabla_{uu}\mathcal{P}(u) & [(\nabla_u c(y_k,u,\xi_k))^* w_k] & 0 \\ \mathrm{bdiag}(w_j\nabla_y c(y_j,u,\xi_j)) & [w_j\nabla_u c(y_j,u,\xi_j)] & 0 & 0 \\ (w\odot H^{yt})^* & 0 & 0 & H^{tt} \end{bmatrix} \begin{bmatrix} \delta_{\vec{y}} \\ \delta_u \\ \delta_{\vec{p}} \\ \delta_t \end{bmatrix} = - \begin{bmatrix} \nabla_{\vec{y}}\mathcal{L} \\ \nabla_u\mathcal{L} \\ \nabla_{\vec{p}}\mathcal{L} \\ \nabla_t\mathcal{L} \end{bmatrix},$$

where $\mathrm{bdiag}(H_j)$ constructs a block-diagonal operator with $H_1, \dots, H_N$ along the diagonal.

4.2 | Gauss–Newton system

Further simplifications of the Hessian can be introduced. First, we can note that all terms corresponding to differentiating $\mathcal{L}$ in y_j or p_j first contain the quadrature weight w_j. In higher dimensions w_j may vary over many orders of magnitude, and keeping them in the Hessian may render it extremely ill-conditioned. Instead, we divide the corresponding rows of the Hessian together with the right hand side (the derivatives of $\mathcal{L}$) by w_j in our formulation directly. This leads to a non-symmetric Gauss–Newton system, but a much lower condition number together with a potent preconditioner developed below allows one to use GMRES or BiCGStab with only a few iterations.

Second, in the case of a linear-quadratic control we have $\mathcal{P}(u) = \frac{1}{2}\langle u, M_u u\rangle$, and $c(y,u,\xi) = \hat{c}(y,\xi) + Bu$. In a weakly nonlinear case we can consider an inexact Newton method, where $\mathcal{P}(u)$ and $c(y,u,\xi)$ are approximated in this form. This allows one to resolve (35) explicitly and plug the corresponding component $u = \mathrm{Proj}_{U_{ad}}((\alpha M_u)^{-1}(-B)^*\sum_{i=1}^N w_i p_i)$ back into the Lagrangian, reducing it to three variables $(\vec{y},\vec{p},t)$. The reduced Gauss-Newton Hessian term appears:

$$\nabla_{p_j,p_k}\mathcal{L} = -w_j B P_r (\alpha M_u)^{-1} B^* w_k, \qquad P_r = \mathrm{Proj}'_{U_{ad}}\left((\alpha M_u)^{-1}(-B)^*\sum_{i=1}^N w_i p_i\right), \tag{37}$$

where w_j in the leftmost position is pending to removal as described above, and $\mathrm{Proj}'_{U_{ad}}$ is a semi-smooth derivative of the projector (e.g. for box constraints this is just an indicator of U_{ad}).

Remark 1. In addition to being quadratic, the control penalty is equivalent to a (weighted) squared L_2-norm in many cases. This renders discretization of M_u spectrally close to a diagonal matrix $\tilde{M}_u$; for example, one may use a standard lumping of the finite element mass matrix. This makes the discretized Hessian (37) easy to assemble, e.g. sparse when M_u is replaced by $\tilde{M}_u$. The case of a H^1-norm control penalty is more limiting, and may require a matrix-free application of M_u^{-1} using, for example, a multigrid method.

The purpose of this elimination of u becomes more apparent for the solution of the Gauss–Newton system in the TT format. An Alternating Least Squares method (cf. Appendix B) tailored to KKT systems[31] requires that all solution components are represented in the same TT decomposition.[32] Since $\vec{y}$ and $\vec{p}$ have the same size, this holds naturally; the only additional variable t is a single number that can be embedded into the same TT decomposition at a little cost. In contrast, a (potentially large) component u needs a nontrivial padding that may inflate the TT ranks and/or make the Hessian more ill-conditioned.

Finally, we obtain the following linear system on the solution increments:

$$\begin{bmatrix} H^{yy} & A^* & H^{yt} \\ A & -\mathcal{B}\otimes W & 0 \\ (H^{yt}_w)^* & 0 & H^{tt} \end{bmatrix} \begin{bmatrix} \delta_{\vec{y}} \\ \delta_{\vec{p}} \\ \delta_t \end{bmatrix} = \begin{bmatrix} F_{\vec{y}} \\ F_{\vec{p}} \\ F_t \end{bmatrix}, \tag{38}$$

where $\mathcal{B} = BP_r(\alpha M_u)^{-1}B^*$, H^{yy} and A are (block) diagonal matrices with H^{yy}_j and $A_j := \nabla_y \hat{c}(y_j, \xi_j)$ on the diagonal,

$$W = \begin{bmatrix} w_1 & \cdots & w_N \\ & \cdots & \\ w_1 & \cdots & w_N \end{bmatrix} \in \mathbb{R}^{N\times N},$$

is a rank-1 matrix (after discretization), and

$$H^{yt} = \begin{bmatrix} H^{yt}_1 \\ \vdots \\ H^{yt}_N \end{bmatrix}, \quad H^{yt}_w = \begin{bmatrix} H^{yt}_1 w_1 \\ \vdots \\ H^{yt}_N w_N \end{bmatrix},$$

whereas the right-hand side components are

$$F_{\vec{y}} = \frac{1}{1-\beta}\left[g'_\varepsilon(\mathcal{J}(y_j,u,\xi_j)-t)\nabla_y \mathcal{J}(y_j,u,\xi_j) + \frac{1}{w_j M}\sum_{\ell=1}^{M} E'_j(\xi_\ell) + (1-\beta)A^*_j p_j \right]_{j=1}^{N},$$

$$F_{\vec{p}} = \left[\hat{c}(y_j,\xi_j) - \mathcal{B}\sum_{i=1}^{N} w_i p_i \right]_{j=1}^{N},$$

$$F_t = 1 - \frac{1}{1-\beta}\left[\sum_{i=1}^{N} w_i g'_\varepsilon(\mathcal{J}(y_i,u,\xi_i)-t) + \frac{1}{M}\sum_{\ell=1}^{M} e'(\xi_\ell) \right].$$

Having solved (38), we perform the Newton update similarly to Algorithm 1, by setting

$$\vec{y}_{m+1} = \vec{y}_m + h\delta_{\vec{y}}, \quad \vec{p}_{m+1} = \vec{p}_m + h\delta_{\vec{p}}, \quad t_{m+1} = t_m + h\delta_t, \quad \text{and} \quad \varepsilon_{m+1} = \mu_\varepsilon \varepsilon_m,$$

where the step size $h > 0$ is chosen ensuring (26) and (27).

Remark 2. Note that (26) is not a proper line search condition and has no theoretical guarantees to lead to a convergent method. However, we have empirically observed in our numerical results that enforcing such a condition allowed our method to converge. A proper line search should be based for example on the Armijo condition applied either to the cost function (for the reduced space formulation in Section 3) or a properly designed merit function (i.e. a weighted sum of the objective function and some norm of the constraint violation) for the full space formulation presented in this section.

Observe that the system (38) can be ill-conditioned and thus requires a good preconditioner to solve it efficiently. In what follows, we discuss a Schur complement-based preconditioner.

4.3 | Preconditioning

In this section we propose a matching strategy[33] to approximate the Schur complement to the Gauss–Newton matrix (38).

First, since δ_t is a single number, we can compute the corresponding Schur complement matrix

$$\begin{bmatrix} H^{yy} - H^{yt}\frac{1}{H^{tt}}(H^{yt}_w)^* & A^* \\ A & -\mathcal{B}\otimes W \end{bmatrix},$$

where we can denote $S^{yy} := H^{yy} - H^{yt}\frac{1}{H^{tt}}(H_w^{yt})^*$ for brevity. Since A is a linearization of $c(y,u,\xi)$ (for example, the PDE operator), it is often invertible. In this case, the Schur complement towards the $(1,2)$ block of this matrix reads

$$S = A^* + S^{yy}A^{-1}(\mathcal{B}\otimes W) = A^*A^{-1}A + S^{yy}A^{-1}(\mathcal{B}\otimes W).$$

We propose a matching approximation consisting of three factors:

$$\tilde{S} = (A^* + \eta S^{yy})A^{-1}\left(A + \frac{1}{\eta}\mathcal{B}\otimes W\right), \tag{39}$$

where $\eta = \sqrt{\|\mathcal{B}\otimes W\|/\|S^{yy}\|}$ is the scaling constant. Note that

$$\|\tilde{S} - S\| = \|\eta S^{yy} + A^*A^{-1}\frac{1}{\eta}\mathcal{B}\otimes W\| = \mathcal{O}(\alpha^{-1/2}),$$

which was shown[31] to be small in norm compared to S for both limits $\alpha\to\infty$ when $\|S\| = \mathcal{O}(1)$, and $\alpha\to 0$ when $\|S\| = \mathcal{O}(\alpha^{-1})$.

Ultimately, we obtain the following right preconditioner:

$$P = \begin{bmatrix} 0 & \tilde{S} & H^{yt} \\ A & -\mathcal{B}\otimes W & 0 \\ 0 & 0 & H^{tt} \end{bmatrix}. \tag{40}$$

Note that this is a permuted block-triangular matrix, solving a linear system with which requires the solution of smaller systems with H^{tt}, $\tilde{S}$ and A. In turn, the solution with $\tilde{S}$ requires the solution with $(A^* + \eta S^{yy})$ and $\left(A + \frac{1}{\eta}\mathcal{B}\otimes W\right)$. If the constraints are defined by a PDE, A, A^* and $g_\varepsilon'(\mathcal{J}-t)\nabla_{yy}\mathcal{J}$ (inside H^{yy}) are sparse matrices, whereas the remaining terms $g_\varepsilon''(\mathcal{J}-t)\nabla_y\mathcal{J}(\nabla_y\mathcal{J})^*$, $H^{yt}\frac{1}{H^{tt}}\left(H_w^{yt}\right)^*$ and W are low-rank matrices, and can be accounted for using the Sherman–Morrison formula.

5 | NUMERICAL RESULTS

In this section, we present various numerical examples to illustrate the efficiency of the proposed approach in both the reduced and full-space formulations. This section in fact goes beyond the above theoretical presentation in multiple ways. In Section 5.1, we consider an optimal control problem in one spatial dimension and random coefficient. We study the approximation error in CVAR_β due to each of the variables ε, n_y (spatial discretization), n_ξ,, d, and TT truncation tolerance. We propose a strategy to select these parameters by equidistribution of the total error. Control variate strategy is applied to this problem in Section 5.2. Section 5.3 focuses on the impact of the quantile β and the ε reduction factor μ_ε. In Section 5.4, we consider the two spatial dimension version of the problem and carry out a comparison between the reduced and full space formulations. We conclude with a realistic problem in Section 5.5, where we propose a risk-averse strategy for lockdown due to pandemics such as COVID-19.

The TTRISK (Algorithm 1) is implemented based on TT-Toolbox[2], whereas for the TT-Cross (Algorithm 2) we use a rank-adaptive implementation `amen_cross_s` from TT-IRT[3]. We run the computations using a default multithreading in Matlab R2019b that can spawn up to 10 threads in BLAS on an Intel Xeon E5-2640 v4 CPU.

5.1 | Elliptic PDE with affine coefficient

We first test the reduced formulation. Consider a PDE in one space dimension with random coefficients κ.

$$\begin{aligned} -\frac{d}{dx}\kappa(x,\xi)\frac{dy}{dx} &= Bu, && x\in(0,1), \quad \xi\sim\mathcal{U}(-\sqrt{3},\sqrt{3})^d, \\ y(-1) = y(1) &= 0, \end{aligned} \tag{41}$$

where $\mathcal{U}(-\sqrt{3}, \sqrt{3})^d$ denotes the uniform distribution on $[-\sqrt{3}, \sqrt{3}]^d$. The KL expansion of $\kappa(x, \xi)$ truncated to d terms,

$$\kappa(x, \xi) = \kappa_0(x) + \sum_{k=1}^{d} \sqrt{\lambda_k}\kappa_k(x) \cdot \xi^{(k)}, \tag{42}$$

is defined by the mean $\kappa_0(x) = 10$ and the eigenvalue decomposition

$$\int C(x, x')\kappa_k(x')dx' = \lambda_k \kappa_k(x), \qquad \lambda_1 \geq \lambda_2 \geq \cdots \geq 0, \tag{43}$$

of the covariance operator with the function

$$C(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right), \qquad \ell = 0.25,$$

for $x, x' \in (0, 1)$, and $\xi^{(k)} \in \mathcal{U}(-\sqrt{3}, \sqrt{3})$, so that $\rho^{(k)}(\xi^{(k)}) = 1/(2\sqrt{3})$.

We use a misfit objective function,

$$\mathcal{J}(y, u; \xi) = \frac{1}{2}\|y(u; x, \xi) - y_d(x)\|^2_{L_2(0,1)}, \tag{44}$$

with the desired state $y_d(x) \equiv 1$. The control u is defined on a subdomain $(0.25, 0.75)$, and B is the identity insertion operator:

$$Bu(x) = \begin{cases} u(x), & x \in (0.25, 0.75), \\ 0, & \text{otherwise.} \end{cases}$$

The PDE is discretized using piecewise linear finite elements on a uniform grid with n_y points $0 = x_1 < \cdots < x_{n_y} = 1$, with the coefficient $\kappa(x, \xi)$ and the control $u(x)$ discretized by collocation at the midpoints $\{x_{i+1/2}\}$. The random variables $\xi^{(1)}, \ldots, \xi^{(d)}$ are discretized by collocation at n_ξ Gauss–Legendre points on the interval $(-\sqrt{3}, \sqrt{3})$.

We aim at estimating the total cost-error scaling. However, since the computation depends on a number of approximation parameters (ε, n_y, n_ξ, d, and the TT truncation tolerance), those need to be selected judiciously to obtain the optimal total complexity. Next, we estimate the error (in CVaR) contributed by each of the parameters. This will allow us to select the parameter values by equalizing their corresponding error estimates.

The (relative) CVaR error at given parameters is defined as

$$\text{err(CVaR)} = \frac{\mathcal{R}^{\varepsilon}_{t_m,\beta,N} - \mathcal{R}_*}{\mathcal{R}_*}, \tag{45}$$

where t_m is the output of Algorithm 1, and $\mathcal{R}_*$ is the reference solution computed at the finest parameters $\varepsilon = 3 \cdot 10^{-4}$, tolerance $= 10^{-5}$, $n_y = 1025$, $n_\xi = 33$, $d = 10$. We take the control regularization parameter $\alpha = 10^{-6}$ and the confidence threshold $\beta = 0.5$. In the following figures, we vary those parameters one by one, keeping the other fixed to those reference values.

Figure 1 shows that the error in CVaR depends almost linearly on ε (in fact the decay is slightly faster, which may be due to particular symmetries in the solution). This is consistent with Reference 26 (lemma 3.4.2) where a linear theoretical convergence was established. However, the TT ranks of the logistic function derivatives grow also linearly with $1/\varepsilon$. This will eventually lead to noticeable computing costs as ε decreases. It is thus crucial that the cross approximation of g''_ε uses a precomputed TT approximation of $j(u, \xi)$ instead of original PDE solutions. In turn, the TT ranks of $j(u, \xi)$ and $\nabla_u j(u, \xi)$ stay almost constant near 40, and hence the number of PDE solutions is almost independent of ε in the considered range.

In Figure 2 we vary the number of quadrature points introduced in each of the random variables ξ. As expected from many previous works (see e.g. References 34-36), the approximation converges exponentially in n_ξ. The TT ranks stabilize towards the value prescribed by ε in Figure 1.

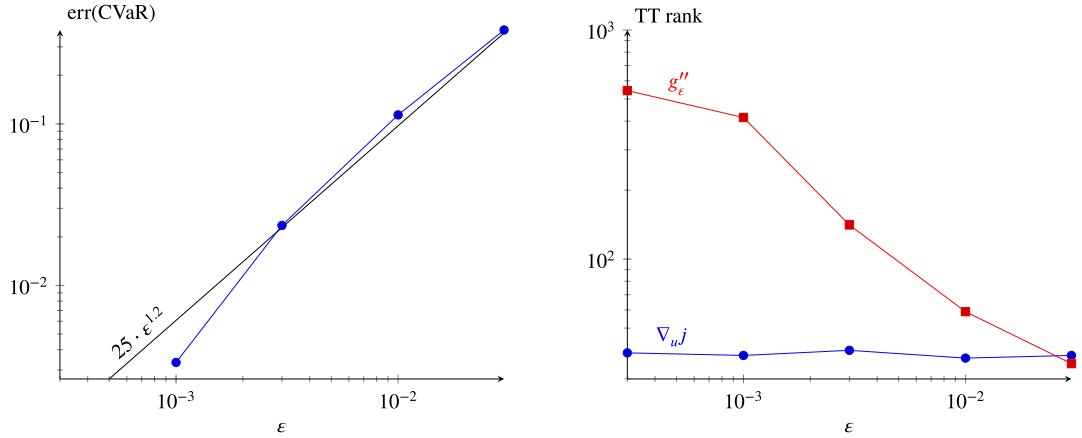

FIGURE 1 Conditional value at risk error (CVaR) (45) and tensor train ranks depending on the CVaR smoothing parameter. Other parameters: tolerance $= 10^{-5}$, $n_y = 1025$, $n_\xi = 33$, $d = 10$, $\alpha = 10^{-6}$, $\beta = 0.5$

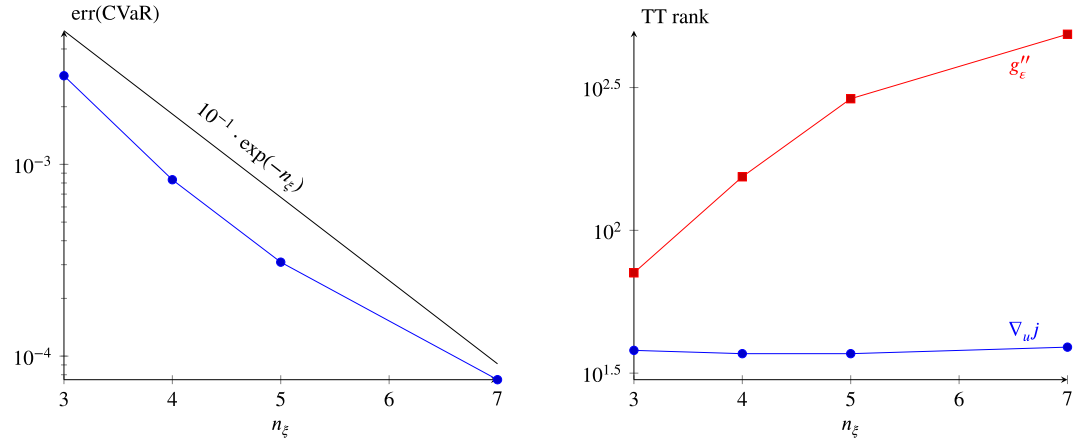

FIGURE 2 Conditional value at risk error error (45) and tensor train ranks depending on the random variable discretization. Other parameters: $\varepsilon = 3 \cdot 10^{-4}$, tolerance $= 10^{-5}$, $n_y = 1025$, $d = 10$, $\alpha = 10^{-6}$, $\beta = 0.5$

Figure 3 benchmarks the scheme against the relative error tolerance used for stopping both the TT-Cross Algorithm 2 and TTRISK Algorithm 1, as well as for the TT approximation. The convergence is linear except for very large values of the threshold, when the TT-Cross may stop prematurely after an accidental drop of the iteration increment below the threshold. The TT ranks grow logarithmically or even slower with 1/tol, which is the enabling observation for many applications of tensor methods.

In Figure 4 we vary the number of finite elements in the spatial discretization of (41). As expected from the second order of consistency of the continuous linear elements, the CVaR error converges quadratically with n_y. The TT ranks stay almost constant, which shows that even the coarsest grid resembles enough qualitative features of the solution.

Lastly in this series, Figure 5 varies the dimension of the random vector ξ, that is, the number of terms in the KL expansion. For the Gaussian covariance matrix of κ we observe the expected exponential convergence in d, and the stabilization of the TT ranks as long as the contribution of the latter random variables $(\xi^{(k)}, \dots, \xi^{(d)}$ for $k \geq k_0$ with some $k_0 > 1)$ becomes negligible compared to the (fixed) TT truncation threshold.

Equipped with the individual error estimates, we can return to estimating the total error-cost scaling. We choose all approximation parameters ($n_y, n_\xi, \varepsilon, d$ and tolerance) such that errors predicted using the rules fitted in Figures 1–5 are equal. Expanding the approximate solution as a Taylor series around the exact solution, we obtain that up to second-order terms, the total error is less than five times the individual error. However, instead of varying directly the total error and calculating all parameters accordingly, it is more convenient to vary the most discrete parameter, to estimate the corresponding error contribution, and to calculate the remaining four parameters using the inverse error prediction rules. The

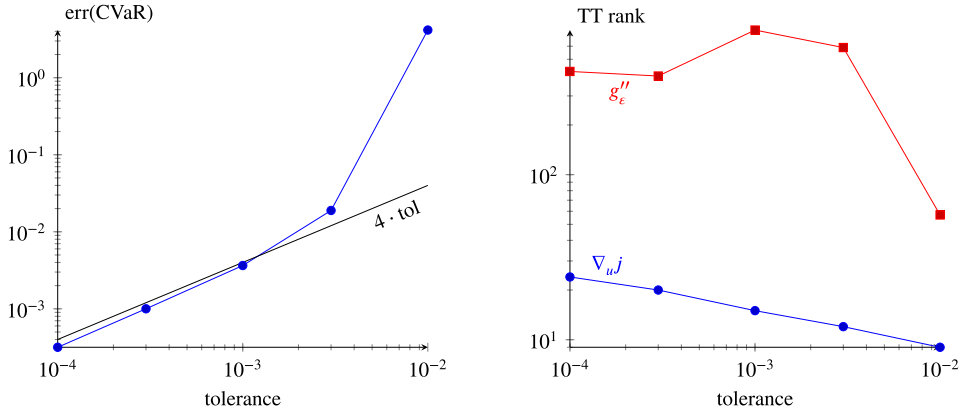

FIGURE 3 Conditional value at risk error error (45) and tensor train (TT) ranks depending on the TT truncation tolerance. Other parameters: $\epsilon = 3 \cdot 10^{-4}$, $n_y = 1025$, $n_\xi = 33$, $d = 10$, $\alpha = 10^{-6}$, $\beta = 0.5$

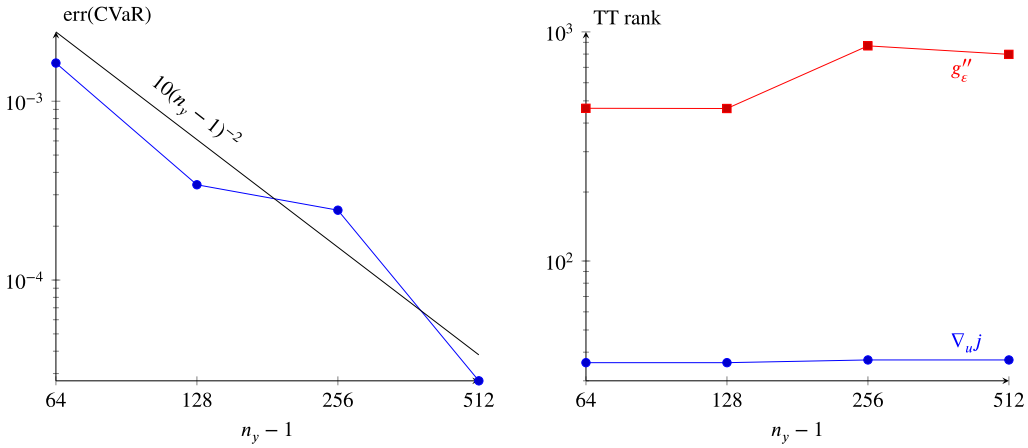

FIGURE 4 Conditional value at risk error (45) and tensor train ranks depending on the spatial discretization. Other parameters: $\epsilon = 3 \cdot 10^{-4}$, tolerance $= 10^{-5}$, $n_\xi = 33$, $d = 10$, $\alpha = 10^{-6}$, $\beta = 0.5$

most discrete parameter is n_y (the spatial grid size), since the number of grid intervals is restricted to a multiple of 4 to ensure that the control subdomain is aligned to all grids considered. For each n_y, this gives the estimated part of the error

$$\frac{\text{error}_{\text{total}}}{5} = 10(n_y - 1)^{-2},$$

and the other parameters are selected as

$$\epsilon = \left(\frac{\text{error}_{\text{total}}}{25 \cdot 5}\right)^{0.83}, \qquad n_\xi = \left\lceil -\log\left(10\frac{\text{error}_{\text{total}}}{5}\right)\right\rceil,$$
$$\text{tol} = \frac{\text{error}_{\text{total}}}{4 \cdot 5}, \qquad d = \left\lceil -\frac{1}{2}\log\left(\frac{\text{error}_{\text{total}}}{5}\right)\right\rceil.$$

In Figure 6 we show the TT ranks, the corresponding numbers of PDE solutions required for the TT-Cross approximation of $j(u, \xi)$ and $\nabla_u j(u, \xi)$, as well as the total computing time, as functions of the *actual* total error for $n_y = 33, 65, 129, 193$ and 257, and the other parameters chosen as above. We observe that both complexity indicators depend linearly on $1/\text{error}$ or slower. This is to be compared with the solution of a deterministic PDE (41) with linear finite elements, which provide an error $= O(n_y^{-2})$ with the computing cost $= O(n_y)$, that is, the deterministic problem scales as cost $= \text{error}^{-1/2}$. We see that the TT solution of a high-dimensional stochastic problem contributes the same amount of complexity.

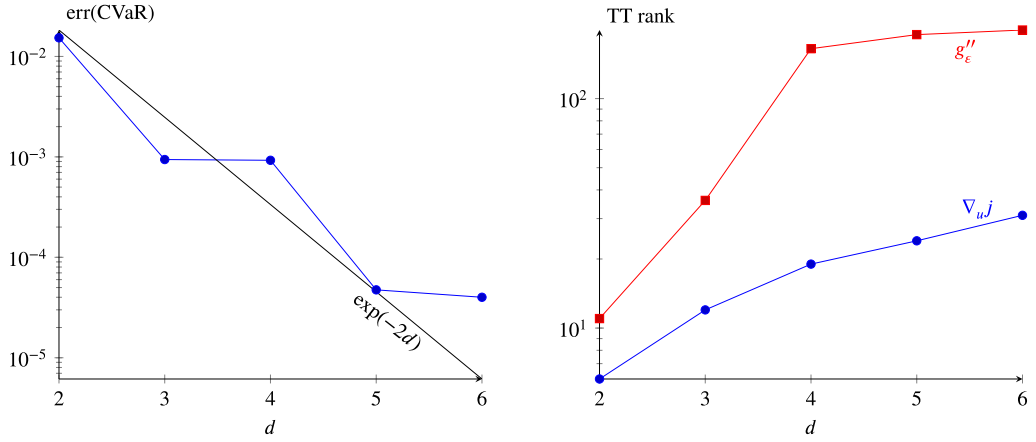

FIGURE 5 Conditional value at risk error (45) and tensor train ranks depending on the number of terms in the Karhunen–Loeve expansion. Other parameters: $\epsilon = 3 \cdot 10^{-4}$, tolerance $= 10^{-5}$, $n_y = 1025$, $n_\xi = 33$, $\alpha = 10^{-6}$, $\beta = 0.5$

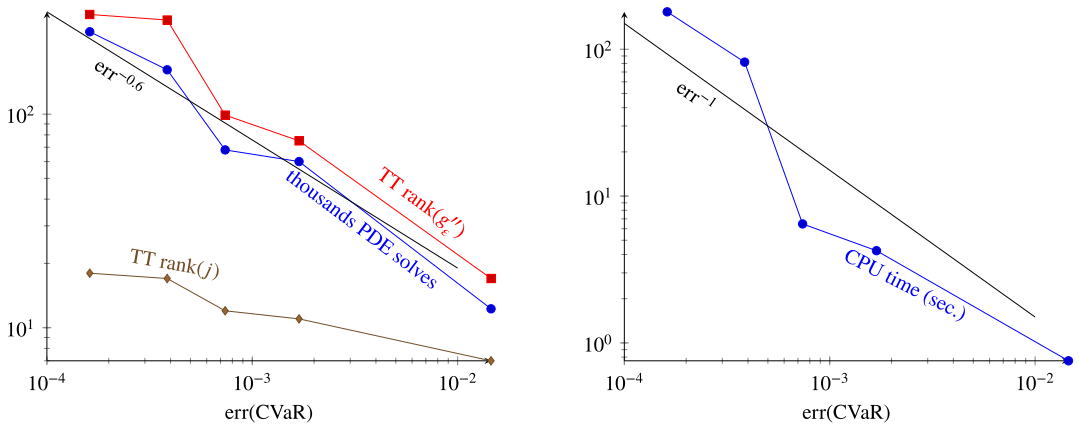

FIGURE 6 Total conditional value at risk error error (45) and computing cost with parameters d, n_y, n_ξ, tol, ϵ chosen to equilibrate their individual error contributions

5.2 | Control variate correction

From Figure 1 we notice that ϵ is the slowest in terms of convergence. In this experiment we add the Monte Carlo control variate correction (33), (34) to the gradient of the CVaR cost function during the optimization, and to the CVaR computation (32) in the end. Note that the first terms in (32)–(34) are deterministic. Therefore, the SDs of (32)–(34) are equal to the SDs of the correction terms only, and can be seen as errors in the Euclidean norm of the underlying probability space. To estimate these errors numerically, we run the algorithm (using some $\tilde{M}$ samples of ξ to compute the averages in (32)–(34)) 16 times, which gives us 16 iid samples of (32)–(34), and compute empirical SDs of those. The ultimate correction is computed as another average of those 16 corrections, so we let $M = 16\tilde{M}$ denote the total number of samples of ξ from all runs of the algorithm. The SD of this ultimate result can be estimated as 1/4th of the empirical SD computed above.

In Figure 7 we show both the ultimate average corrections and standard deviations of the corrections estimated as above. We see that the SDs decay with the law of large numbers, as expected. Moreover, both means and SDs decrease linearly with ϵ.

In multilevel Monte Carlo, the estimated SD can also be used to adapt the number of Monte Carlo samples toward the desired error threshold by extrapolating the law of large numbers.[30] However, the convergence (and error estimation) of the TT approximation can be more complicated. Therefore, we suggest to decrease ϵ until the TT ranks are still manageable, and then compute the control variate correction to both estimate and improve the error. This also allows us to

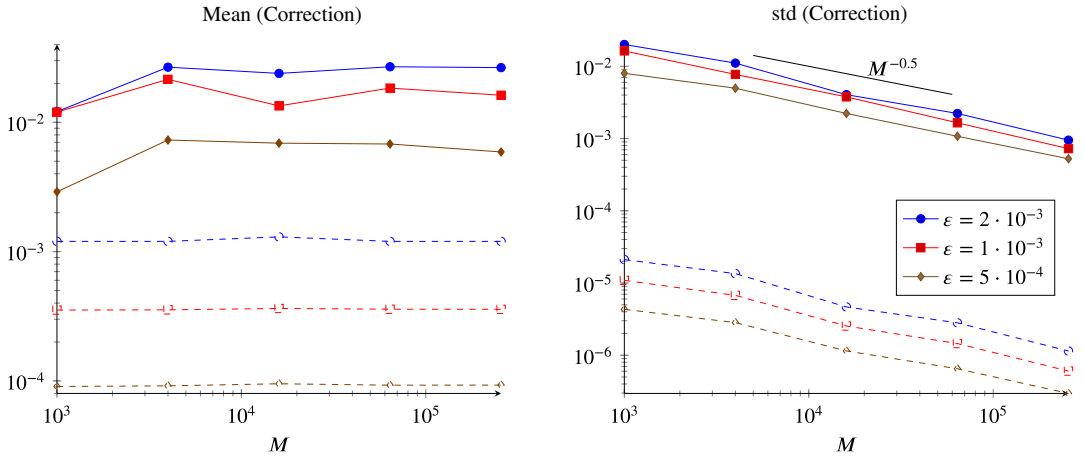

FIGURE 7 Solid lines: average (left) and estimated SD (right) of the correction to the cost gradient (33) and (34) depending on the number of Monte Carlo samples M. Dashed lines: average and SD of the correction to $\mathcal{R}_t$ (32). Spatial discretization $n_y = 129$, other parameters are chosen to equilibrate their individual error contributions.

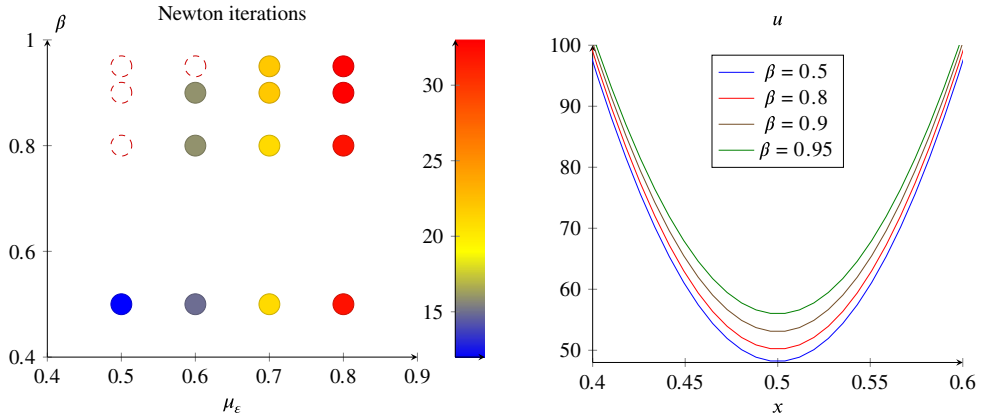

FIGURE 8 Left: number of Newton iterations for different quantiles β and rate of decrease of ε denoted by μ_ε. Right: control signals for different β. Spatial discretization $n_y = 129$, other parameters are chosen to equilibrate their individual error contributions.

decouple the TT and Monte Carlo steps, or to reuse previously computed TT approximations of, for example, the forward model solution.

5.3 | Effect of the quantile

In the last experiment with the 1D PDE, we vary the quantile of the confidence interval in CVaR. The results are reported in Figure 8. As β increases, the model minimizes the cost g with higher confidence, which requires a stronger control (see the right plot of Figure 8). However, the increasing $1/(1-\beta)$ term in the gradient and Hessian renders the Newton method slower and less reliable. Recall that we start with a large ε and decrease it geometrically with Newton iterations according to (25). This allows us to avoid getting too small values of g''_ε. The default value used in the previous experiments was $\mu_\varepsilon = 0.5$, which was a reasonable balance between the stability of the method and its convergence speed. However, for larger β the Newton method may break at exact machine zeros in g''_ε, leading to infinities in the solution increment. In the left plot of Figure 8 we vary both β and μ_ε independently. Dashed circles denote the experiments where the Newton method failed to converge. We see that larger β requires slower iterations with larger μ_ε. Of course, too large μ_ε will just lead to unnecessarily many iterations. This means that the parameter μ_ε may need problem-dependent tuning, although we believe that the observations in Figure 8 should serve as a good initial guess in a variety of cases.

5.4 | Elliptic PDE with log-normal coefficient

Now we consider a larger problem: we solve a PDE on a two-dimensional space,

$$\begin{aligned} -\nabla_x \cdot \kappa(x,\xi)\nabla_x y &= Bu, \qquad && x \in (0,1)^2, \quad \xi \sim \mathcal{N}(0,\sigma^2 I), \\ y|_{\partial(0,1)^2} &= 0, \end{aligned} \tag{46}$$

where the coefficient is a log-normal random field. Namely, we assume that

$$\log \kappa(x,\xi) = \sum_{k=1}^{d} \sqrt{\lambda_k}\kappa_k(x)\cdot \xi^{(k)},$$

where $\xi_k \sim \mathcal{N}(0,1)$, the standard normally distributed random variable with $\rho^{(k)}(\xi^{(k)}) = \exp(-(\xi^{(k)})^2/2)/\sqrt{2\pi}$, is a KL expansion with mean zero and coefficients λ_k, κ_k defined by the eigenvalue decomposition (43) defined by the covariance function

$$C(x,x') = \sigma^2 \exp\left(-\frac{\|x-x'\|_2^2}{\ell^2}\right), \qquad \ell = 0.5, \quad \sigma^2 = 0.05,$$

and parametrize $\log\kappa(x,\xi)$ with $d = 10$ terms of the KL expansion. This accounts for 98% of the variance. The objective function is similar to (44),

$$\mathcal{J}(y,u;\xi) = \frac{1}{2}\|y(u;x,\xi) - y_d(x)\|^2_{L_2((0,1)^2)}, \qquad y_d(x) \equiv 1. \tag{47}$$

The control u is defined on a disk inside the domain, and B is the identity insertion operator:

$$Bu(x) = \begin{cases} u(x), & \|x - [0.5,0.5]\|_2 \le 0.25, \\ 0, & \text{otherwise.} \end{cases}$$

This gives a challenging enough case of incomplete control. The PDE is discretized using piecewise linear finite elements on a triangular grid with 1829 nodes, which is shown in Figure 9.

The random variables $\xi^{(1)}, \dots, \xi^{(d)}$ are discretized by collocation at $n_\xi = 9$ Gauss–Hermite points.

In this test we set $\beta = 0.8$, and correspondingly $\mu_\varepsilon = 0.7$ (cf. Figure 8). The final smoothness width $\varepsilon = 3\cdot 10^{-3}$, consistent with the TT approximation error threshold $\text{tol} = 10^{-3}$, the discretization, and KL truncation errors. In Figure 10 we vary the control regularization parameter α, and compare the reduced formulation introduced in Section 3, and the Lagrangian formulation from Section 4.

We see that the TT ranks of derivatives of the sigmoid function g_ε in both formulations are of the same scale. However, since the forward model involves solving a PDE, the bottleneck is actually the computation of a surrogate of the PDE solution: the TT approximation cost function $j(u;\xi)$ in the reduced formulation, and the block TT format of $\delta_{\bar{y}}$ and $\delta_{\bar{p}}$ in the Lagrangian formulation. We see that the TT ranks of the Lagrangian solution are 50% larger than those of j, which is actually lower than a factor of 2 expected for a TT format representing two components simultaneously ($\delta_{\bar{y}}$ and $\delta_{\bar{p}}$). We see also that the TT ranks of j, $\delta_{\bar{y}}$ and $\delta_{\bar{p}}$ depend little on α.

Similarly, the number of Newton iterations in the Lagrangian formulation stays nearly constant, whereas in the reduced formulation the number of iterations grows with $\log\alpha$. This is due to the KKT matrix being the exact Hessian of the Lagrangian, whereas the reduced formulation uses only an approximate (fixed point) Hessian of $j(u,\xi)$. This is even more crucial for a large β as we have also seen in the previous test. The imperfection of the reduced Hessian can be seen in the growing number of Newton iterations. However, if we look at CPU times (Figure 10 right), we notice that the time of the Lagrangian method grows rapidly with α despite nearly constant number of Newton iterations and TT ranks. This is due to the growth of GMRES iterations in solving Equation (38). The condition number of the KKT matrix preconditioned with Equation (40) may still deteriorate with α in the considered case of a control on a subdomain. A better preconditioner may reduce the complexity. Nevertheless, for moderate values of α even with the current preconditioner the Lagrangian formulation is preferable to the reduced one.

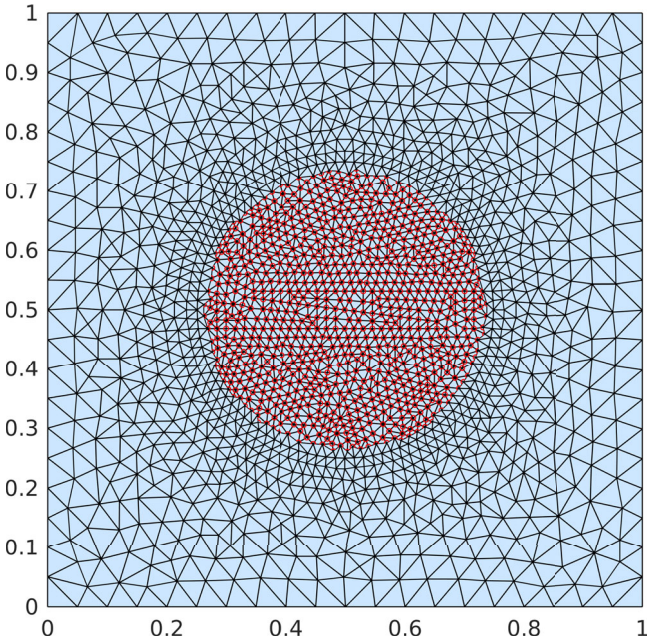

FIGURE 9 Mesh used to discretize Equation (46). Red points denote grid nodes belonging to the control.

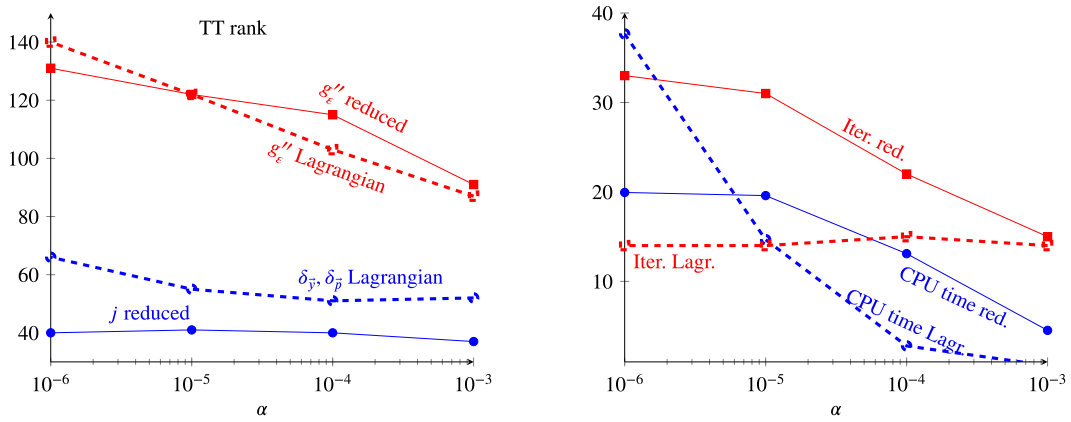

FIGURE 10 Tensor train ranks (left), CPU times (hours) and numbers of Newton iterations (right) in the reduced and Lagrangian TTRISK formulations depending on the control regularization parameter α (as defined in (1)) in the two-dimensional partial differential equation problem.

5.5 | Infection ODE model

In the last example, we experiment with an epidemiological ODE model from Reference 1, which was used to estimate the progression of COVID-19 in the United Kingdom for 90 days starting from the March 1, 2020. The model considers population dynamics split into the following compartments:

- Susceptible (S): individuals who are not in contact with the virus at the moment.
- Exposed (E) to the virus, but not yet infectious.
- Infected SubClinical (I^{SC1}) at the moment, but who may require hospitalizations.
- Infected SubClinical (I^{SC2}), but recovering without any medical intervention.
- Infected Clinical (I^{C1}), individuals in the hospital, who may decease after some time.
- Infected Clinical (I^{C2}) in the hospital, but recovering.

- Recovered (R) and immune to reinfections.
- Deceased (D).

Each of those categories is further stratified into five age groups: 0–19, 20–39, 40–59, 60–79, and 80+. The age group is denoted by a subscript, for example, E_i is the number of exposed individuals in the ith age group ($i = 1, \dots, 5$), and similarly for other compartments. Each variable of the list above, such as $E = (E_1, \dots, E_5)$, is thus a vector of length 5.

On the other hand, three simplifications are in order. First, in the early stage of the pandemic the number of individuals affected by the virus is a small fraction of the entire population. This allows us to take S constant equal to the initial population, and exclude it from the ODE system. Similarly, R and D are terminal states in the sense that none of the other variables depend on them. This again allows us to decouple R and D from the system of ODEs. Thus, we arrive at a linear system involving only Exposed and Infected numbers:

$$\frac{d}{d\tau}\begin{bmatrix} E \\ I^{SC1} \\ I^{SC2} \\ I^{C1} \\ I^{C2} \end{bmatrix} - \begin{bmatrix} -\kappa \mathtt{I} & A_u & A_u & 0 & 0 \\ \kappa \cdot \mathrm{diag}(\rho) & -\gamma_C \mathtt{I} & 0 & 0 & 0 \\ \kappa \cdot \mathrm{diag}(1-\rho) & 0 & -\gamma_R \mathtt{I} & 0 & 0 \\ 0 & \gamma_C \cdot \mathrm{diag}(\rho') & 0 & -\nu \mathtt{I} & 0 \\ 0 & \gamma_C \cdot \mathrm{diag}(1-\rho') & 0 & 0 & -\gamma_{R,C} \mathtt{I} \end{bmatrix} \begin{bmatrix} E \\ I^{SC1} \\ I^{SC2} \\ I^{C1} \\ I^{C2} \end{bmatrix} = 0. \tag{48}$$

Here $\mathtt{I}$ is the identity matrix of size 5, diag$(\cdot)$ stretches a vector into a diagonal matrix, $A_u = \chi \cdot \mathrm{diag}(S) \cdot C_u \cdot \mathrm{diag}\left(\frac{1}{N}\right)$, and the remaining variables are model parameters:

- χ: probability of contact between S and I^{SC} individuals.
- $\kappa = 1/d_L$: transition rate of Exposed becoming SubClinical. d_L is the number of days in the Exposed state.
- $\gamma_C = 1/d_C$: transition rate of SubClinical turning into Clinical. Similarly, d_C is the number of days this transition takes.
- $\gamma_R = 1/d_R$: recovery rate from I^{SC2}.
- $\gamma_{R,C} = 1/d_{R,C}$: recovery rate from I^{C2}.
- $\nu = 1/d_D$: death rate from I^{C1}.
- $\rho = (\rho_1, \dots, \rho_5)^\top \in \mathbb{R}^5$: age-dependent probabilities of Exposed turning into the first SubClinical category.
- $\rho' = (\rho'_1, \dots, \rho'_5)^\top \in \mathbb{R}^5$: age-dependent probabilities of SubClinical becoming the first Clinical category.
- $N = (N_1, \dots, N_5)^\top \in \mathbb{R}^5$: population sizes in each age group.
- $C_u \in \mathbb{R}^{5\times 5}$: the contact matrix, which depends on the control u.

Scalar-vector operations ($1/N$, $1-\rho$, etc.) are understood as elementwise operations.

Another parameter is the number of infected individuals on day 0 (1 March) N^0. It is split further across the age groups as follows:

$$N^{in} := \left(N_1^{in}, N_2^{in}, N_3^{in}, N_4^{in}, N_5^{in}\right)^\top = (0.1, 0.4, 0.35, 0.1, 0.05)^\top N^0.$$

The ODE (48) is initialized by setting

$$E(0) = \frac{N^{in}}{3}, \quad I^{SC1}(0) = \frac{2}{3}\mathrm{diag}(\rho)N^{in}, \quad I^{SC2}(0) = \frac{2}{3}\mathrm{diag}(1-\rho)N^{in}, \quad I^{C1}(0) = I^{C2}(0) = 0.$$

The population size $S = N$ is taken from the office of national statistics, mid 2018 estimate.

The contact matrix consists of four contributions, corresponding to people activities[4]:

$$C_u = \mathrm{diag}(\alpha^{\mathrm{home}})C^{\mathrm{home}} + \mathrm{diag}(\alpha^{\mathrm{work}})C^{\mathrm{work}} + \mathrm{diag}(\alpha^{\mathrm{school}})C^{\mathrm{school}} + \mathrm{diag}(\alpha^{\mathrm{other}})C^{\mathrm{other}}, \tag{49}$$

where C^* are pre-pandemic contact matrices (for details of their estimation see Reference 1), and α^* are coefficients of reduction of activities introduced in March 2020. (Here $*$ stands for home, work, school, or other.) In turn, those are constructed as follows. Firstly, we set $\alpha^{\text{home}} = (1, \dots, 1)$, since home contacts cannot be influenced. For the remaining activities, noting that the lockdown in the UK was called on day 17 (18th March), we set

$$\alpha^*(\tau) = \begin{cases} (1,1,1,1,1)^\top, & \tau < 17, \\ (\alpha_{123}(1-u^*(\tau)), \alpha_{123}(1-u^*(\tau)), \alpha_{123}(1-u^*(\tau)), \alpha_4, \alpha_5)^\top, & \text{otherwise}, \end{cases} \tag{50}$$

where u^{work}, u^{school} and u^{other} are the control signals (the lockdown measures) that we are going to optimize, and $\alpha_{123}, \alpha_4, \alpha_5$ are their proportions in the corresponding age groups.

In the cost function we penalize the total fatalities and the hospital capacity exceedance. As long as (48) is solved, the number of deaths can be calculated directly as

$$D(\tau) = \nu \int_0^\tau I^{C1}(s) ds. \tag{51}$$

Moreover, we aggregate $I^C = \sum_{i=1}^5 I_i^{C1} + I_i^{C2}$ and penalize I^C exceeding 10,000. Finally, we penalize the strength of the lockdown measures, that is, the norm of the control $u(\tau) = (u^{\text{work}}(\tau), u^{\text{school}}(\tau), u^{\text{other}}(\tau))$, as well as constraining $u^{\text{work}} \in [0, 0.69]$, $u^{\text{school}} \in [0, 0.9]$ and $u^{\text{other}} \in [0, 0.59]$. This gives us a deterministic cost

$$\mathcal{J}^{Det}(u) = \frac{1}{2}\left[D(T) + \int_0^T \max(I^C(\tau) - 10{,}000, 0) d\tau + \zeta \int_{17}^T \|u(\tau)\|_2^2 d\tau\right], \tag{52}$$

where $T = 90$ is the simulation interval, and ζ is the regularization parameter.

However, optimization of (52) may be misleading, since the model parameters are not known in advance, and can only be estimated. In particular, Reference 1 employed an Approximate Bayesian Computation (ABC), which used existing observations of daily deaths and hospitalizations in the United Kingdom to form the likelihood, and consequently the posterior probability density function. This renders model parameters into random variables, which are distributed according to the posterior density. In turn, this motivates a modification of (52) into a risk-averse cost function, for example, using CVaR with $\beta = 0.5$. More precisely, we aim to minimize

$$\mathcal{J}^{CVaR}(u) = \text{CVaR}_{0.5}^{\epsilon}\left(\frac{1}{2}D(T) + \frac{1}{2}\int_0^T \max(I^C(\tau) - 10{,}000, 0) d\tau\right) + \frac{\zeta}{2}\int_{17}^T \|u(\tau)\|_2^2 d\tau. \tag{53}$$

Ideally, the expectations in CVaR need to be computed with respect to the posterior density from ABC. However, the latter is a complicated multivariate function, which lacks an independent variable parametrization necessary to set up the discretization and TT approximation. (A possible solution to this using optimal transport[37] can be a matter of future research.) As a proof of concept, we simplify the distribution to independent uniform, reflecting means and variances estimated by ABC. Thus, we assume

$$\begin{aligned}
\chi &\sim \mathcal{U}(0.13 - 0.03\sigma, 0.13 + 0.03\sigma), & d_L &\sim \mathcal{U}(1.57 - 0.42\sigma, 1.57 + 0.42\sigma), \\
d_C &\sim \mathcal{U}(2.12 - 0.80\sigma, 2.12 + 0.80\sigma), & d_R &\sim \mathcal{U}(1.54 - 0.40\sigma, 1.54 + 0.40\sigma), \\
d_{R,C} &\sim \mathcal{U}(12.08 - 1.51\sigma, 12.08 + 1.51\sigma), & d_D &\sim \mathcal{U}(5.54 - 2.19\sigma, 5.54 + 2.19\sigma), \\
\rho_1 &\sim \mathcal{U}(0.06 - 0.03\sigma, 0.06 + 0.03\sigma), & \rho_2 &\sim \mathcal{U}(0.05 - 0.03\sigma, 0.05 + 0.03\sigma), \\
\rho_3 &\sim \mathcal{U}(0.08 - 0.04\sigma, 0.08 + 0.04\sigma), & \rho_4 &\sim \mathcal{U}(0.54 - 0.22\sigma, 0.54 + 0.22\sigma), \\
\rho_5 &\sim \mathcal{U}(0.79 - 0.14\sigma, 0.79 + 0.14\sigma), & \rho_1' &\sim \mathcal{U}(0.26 - 0.23\sigma, 0.26 + 0.23\sigma), \\
\rho_2' &\sim \mathcal{U}(0.28 - 0.25\sigma, 0.28 + 0.25\sigma), & \rho_3' &\sim \mathcal{U}(0.33 - 0.27\sigma, 0.33 + 0.27\sigma), \\
\rho_4' &\sim \mathcal{U}(0.26 - 0.11\sigma, 0.26 + 0.11\sigma), & \rho_5' &\sim \mathcal{U}(0.80 - 0.13\sigma, 0.80 + 0.13\sigma), \\
N^0 &\sim \mathcal{U}(276 - 133\sigma, 276 + 133\sigma), & \alpha_{123} &\sim \mathcal{U}(0.63 - 0.21\sigma, 0.63 + 0.21\sigma), \\
\alpha_4 &\sim \mathcal{U}(0.57 - 0.23\sigma, 0.57 + 0.23\sigma), & \alpha_5 &\sim \mathcal{U}(0.71 - 0.23\sigma, 0.71 + 0.23\sigma),
\end{aligned} \tag{54}$$

TABLE 1 Conditional value at risk (CVaR) optimizer behavior for different variability of random variables

σ	$\mathcal{J}^{\text{CVaR}}$	$\frac{\zeta}{2}\int_{17}^{T}\|u(\tau)\|_2^2 d\tau$	t	Iterations	Tensor train rank ($\nabla_u \mathcal{J}^{\text{CVaR}}$)
10^{-2}	171,168	3107	158,382	16	13
10^{-1}	289,160	3439	157,750	20	68

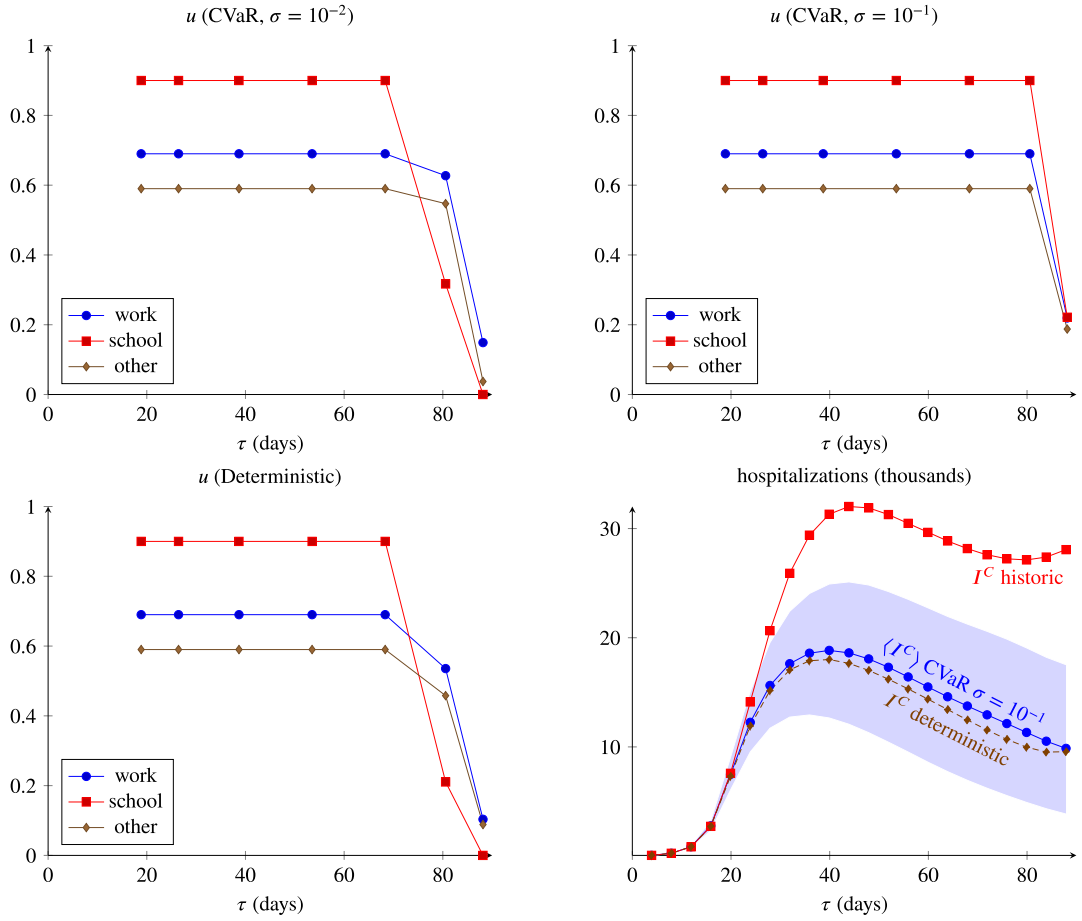

FIGURE 11 Control signals and predicted hospitalizations numbers for different optimization strategies. In the bottom right plot, blue circles denote mean I^C after conditional value at risk optimization, shaded area denotes 95% confidence interval.

where σ is a variance tuning parameter. That is, (54) form a random vector

$$\xi = (\chi, d_L, d_C, d_R, d_{R,C}, d_D, \rho_1, \rho_2, \rho_3\rho_4, \rho_5, \rho_1', \rho_2', \rho_3'\rho_4', \rho_5', N^0, \alpha_{123}, \alpha_4, \alpha_5),$$

of dimension $d = 20$, the state vector is

$$y = (E_1, \dots, E_5,\ I_1^{SC1}, \dots, I_5^{SC1},\ I_1^{SC2}, \dots, I_5^{SC2},\ I_1^{C1}, \dots, I_5^{C1},\ I_1^{C2}, \dots, I_5^{C2}),$$

and the left-hand side of (48) constitutes the constraints c.

For the deterministic optimization (52) we set all variables to their means. For the stochastic optimization (53) we discretize each variable by a 3-point Gauss–Legendre quadrature rule on the corresponding interval, and the TT approximations are computed with relative error threshold of 10^{-2}. The control is discretized with a Gauss–Lagrange interpolation on $[17, T]$ with 7 points, which makes the total dimension of the discretized control space 21. This allows us to use the reduced optimization formulation. Moreover, since the cost contains nonsmooth functions, we abandon the Newton scheme, resorting to the Projected Gradient Descent method with a finite difference computation of the gradient

of (53). The ODE (48) is solved using the implicit Euler method with time step 0.1. The control regularization $\zeta = 100$ is taken from,[1] and the CVaR smoothing parameter $\epsilon = 1000$ corresponds to the relative width of the smoothed region $\epsilon/t < 10^{-2}$, matching the bias of the smoothed CVaR and the TT approximation error.

In Table 1 we vary σ and investigate the behavior of the reduced TT formulation for CVaR. Note that we need more iterations and higher TT ranks for the larger variance. This is also reflected in a larger total cost, which is dominated by the CVaR term. In Figure 11, we compare the controls computed with the two values of σ, as well as the minimizer of the deterministic cost (52). We see that a small σ yields the solution which is similar to the deterministic one. However, the risk-averse control for a larger variance of the parameters is more conservative: it tends to be larger, hitting the constraints at almost the entire interval except the final point, where the control stops making any influence on the system.

In the bottom right panel of Figure 11 we plot the predicted hospitalization numbers. The historic numbers are obtained by simulating the deterministic model (using mean values of the parameters in (54)) with the control derived from the smoothed Google daily mobility data[5]. We see that both optimization techniques allow one to reduce the hospital occupancy. However, the deterministic approach tends to underestimate I^C compared to the mean risk-averse value. In addition, the CVaR frameworks enables a rigorous uncertainty quantification.

6 | CONCLUSION AND OUTLOOK

This paper has introduced a new algorithm called TTRISK to solve risk-averse optimization problems constrained by differential equations (PDEs or ODEs). TTRISK can be applied to both the reduced and full-space formulations. The article also introduces a control variate correction to get unbiased estimators. Various strategies to choose the underlying algorithmic parameters have been outlined throughout the paper, especially in the numerics section. This is carried out by carefully taking into account all the approximation errors.

The TT framework offers multiple advantages, for instance, our numerical examples illustrate that it can help overcome the so-called "curse of dimensionality." Indeed, the approach introduced here has been successfully applied to a realistic problem, with 20 random variables, to study the propagation of COVID-19 and to devise optimal lockdown strategies.

This article aims to initiate new research directions in the field of risk-averse optimization. There are many open questions that one could pursue, a few examples are:

1. Convergence analysis of TTRISK in both reduced and full-space formulations.
2. Convergence analysis of TTRISK in the presence of control variate terms.
3. Convergence analysis of preconditioned Gauss-Newton method (cf. Section 4.3) for the full-space formulation.
4. The Gauss-Newton system (cf. Section 4.2) and preconditioning (cf. Section 4.3) for the full space formulation eliminates the control and it considers a formulation in terms of y, p, and t. It maybe interesting to design preconditioners which can handle control u directly in the full-space formulation.

ACKNOWLEDGMENTS

Harbir Antil and Akwum Onwunta are partially supported by NSF grants DMS-2110263, DMS-1913004 and the Air Force Office of Scientific Research under Award number: FA9550-19-1-0036 and FA9550-22-1-0248. Sergey Dolgov is thankful for the support from Engineering and Physical Sciences Research Council (EPSRC) New Investigator Award EP/T031255/1 and New Horizons grant EP/V04771X/1.

CONFLICT OF INTEREST

This study does not have any conflicts to disclose.

DATA AVAILABILITY STATEMENT

Matlab codes for reproducing the experiments and the data that support the findings of this study are openly available in TTRISK at https://github.com/dolgov/TTRISK.

ENDNOTES

[1]See Remark 2 for further explanation of this condition.

[2]https://github.com/oseledets/TT-Toolbox

[3]https://github.com/dolgov/TT-IRT

[4]Note that in this section α is the notation from the original paper,[1] not a regularization parameter.

[5]The historic I^C differs from that in Reference 1 (figure 6). This is most likely due to imprecise reproduction of the parameters. However, the values agree within a factor of 2, which indicates that the qualitative behavior of the model is correct.

REFERENCES

1. Dutta R, Gomes SN, Kalise D, Pacchiardi L. Using mobility data in the design of optimal lockdown strategies for the COVID-19 pandemic. PLoS Comput Biol. 2021;17(8):1–25.
2. Durlofsky LJ, Chen Y. Uncertainty quantification for subsurface flow problems using coarse-scale models. In: Graham I, Hou T, Lakkis O, Scheichl R, (eds). Numerical analysis of multiscale problems. Berlin, Heidelberg: Springer; 2012. p. 163–202.
3. Petrat N, Zhu H, Stadler G, Hughes TJ, Ghattas O. An inexact Gauss-Newton method for inversion of basal sliding and rheology parameters in a nonlinear stokes ice sheet model. J Glaciol. 2012;58(211):889–903.
4. Tartakovsky DM, Guadagnini A, Riva M. Stochastic averaging of nonlinear flows in heterogeneous porous media. J Fluid Mech. 2003;492:47–62. https://doi.org/10.1017/S002211200300538X
5. Kouri DP, Surowiec TM. Risk-averse optimal control of semilinear elliptic PDEs. ESAIM Control Optim Calc Var. 2020;26:53. https://doi.org/10.1051/cocv/2019061
6. Kouri DP, Surowiec TM. Risk-averse PDE-constrained optimization using the conditional value-at-risk. SIAM J Optim. 2016;26(1):365–96. doi:10.1137/140954556
7. Kouri DP, Shaprio A. Optimization of PDEs with uncertain inputs. In: Antil H, Kouri DP, Lacasse MD, Ridzal D, editors. Frontiers in PDE-constrained optimization. Volume 163. Berlin, Heidelberg, New-York: Springer Verlag; 2018. p. 41–81.
8. Garreis S, Surowiec TM, Ulbrich M. An interior-point approach for solving risk-averse PDE-constrained optimization problems with coherent risk measures. SIAM J Optim. 2021;31(1):1–29. https://doi.org/10.1137/19M125039X
9. Rockafellar RT, Uryasev S. Optimization of conditional value-at-risk. J Risk. 2000;2:21–41.
10. Shapiro A, Dentcheva D, Ruszczyński A. Lectures on stochastic programming. Volume 9 of MOS-SIAM series on optimization. 2nd ed. Philadelphia, PA: SIAM; 2014 Modeling and theory.
11. Rockafellar RT, Uryasev S. Conditional value-at-risk for general loss distributions. J Bank Finance. 2002;26:1443–71.
12. Kouri D. An approach for the adaptive solution of optimization problems governed by partial differential equations with uncertain coefficients [PhD thesis]. Rice University; 2012.
13. Kouri DP, Surowiec TM. A primal–dual algorithm for risk minimization. Math Dent Prog. 2022;193:337–363.
14. Hackbusch W. Tensor spaces and numerical tensor calculus. Berlin: Springer–Verlag; 2012.
15. Khoromskij BN. Tensor numerical methods in scientific computing. Berlin: De Gruyter; 2018.
16. Bigoni D, Engsig-Karup AP, Marzouk YM. Spectral tensor-train decomposition. SIAM J Sci Comput. 2016;38(4):A2405–39. https://doi.org/10.1137/15M1036919
17. Gorodetsky A, Karaman S, Marzouk Y. A continuous analogue of the tensor-train decomposition. Comput Methods Appl Mech Eng. 2019;347:59–84. https://doi.org/10.1016/j.cma.2018.12.015
18. Hackbusch W, Khoromskij BN. Low-rank Kronecker-product approximation to multi-dimensional nonlocal operators. I. Separable approximation of multi-variate functions. Comput Secur. 2006;76(3–4):177–202.
19. Schneider R, Uschmajew A. Approximation rates for the hierarchical tensor format in periodic Sobolev spaces. J Complexity. 2013;30(2):56–71.
20. Oseledets IV. Tensor train decomposition. SIAM J Sci Comp. 2011;33(5):2295–317.
21. Oseledets IV, Tyrtyshnikov EE. TT-cross approximation for multidimensional arrays. Linear Algebra Appl. 2010;432(1):70–88.
22. Savostyanov DV, Oseledets IV. Fast adaptive interpolation of multi-dimensional arrays in tensor train format. Proceedings of 7th International Workshop on Multidimensional Systems (nDS); 2011.
23. Dolgov S, Savostyanov D. Parallel cross interpolation for high–precision calculation of high–dimensional integrals. Comput Phys Commun. 2020;246:106869.
24. Giles MB. Multilevel Monte Carlo methods. Acta Numer. 2015;24:259–328.
25. Robert CP, Casella G, Casella G. Monte Carlo statistical methods. Vol 2. New York: Springer; 2004.
26. Markowski M. Newton-based methods for smoothed risk-averse PDE-constrained optimization problems [Masters thesis]. Rice University; 2019.
27. Forsgren A, Gill PE, Griffin JD. Iterative solution of augmented systems arising in interior methods. SIAM J Optim. 2007;18(2):666–90.
28. Bertsekas DP, Hager W, Mangasarian O. Nonlinear programming. Belmont, MA: Athena Scientific; 1999.
29. Nobile F, Tesei F. A multi level Monte Carlo method with control variate for elliptic PDEs with log-normal coefficients. Stoch PDEs Anal Comput. 2015;3(3):398–444.
30. Kuo F, Scheichl R, Schwab C, Sloan I, Ullmann E. Multilevel quasi-Monte Carlo methods for lognormal diffusion problems. Math Comput. 2017;86:2827–60.
31. Benner P, Dolgov S, Onwunta A, Stoll M. Low-rank solvers for unsteady stokes–brinkman optimal control problem with random data. Comput Method Appl M. 2016;304:26–54.
32. Dolgov SV, Khoromskij BN, Oseledets IV, Savostyanov DV. Computation of extreme eigenvalues in higher dimensions using block tensor train format. Comput Phys Commun. 2014;185(4):1207–16.

33. Pearson JW, Stoll M, Wathen AJ. Regularization-robust preconditioners for time-dependent PDE-constrained optimization problems. SIAM J Matrix Anal Appl. 2012;33(4):1126–52.
34. Khoromskij BN, Schwab C. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. SIAM J Sci Comput. 2011;33(1):364–85. https://doi.org/10.1137/100785715
35. Cohen A, Devore R, Schwab C. Analytic regularity and polynomial approximation of parametric and stochastic elliptic PDE's. Anal Appl. 2011;9(1):11–47. https://doi.org/10.1142/S0219530511001728
36. Herrmann L, Schwab C, Zech J. Deep neural network expression of posterior expectations in Bayesian PDE inversion. Inverse Probl. 2020;36(12):125011. https://doi.org/10.1088/1361-6420/abaf64
37. Cui T, Dolgov S. Deep composition of tensor-trains using squared inverse rosenblatt transports. Found Comput Math. 2022;22:1863–1922.
38. Holtz S, Rohwedder T, Schneider R. The alternating linear scheme for tensor optimization in the tensor train format. SIAM J Sci Comput. 2012;34(2):A683–713. https://doi.org/10.1137/100818893
39. White SR. Density matrix algorithms for quantum renormalization groups. Phys Rev B. 1993;48(14):10345–56.
40. Schollwöck U. The density matrix renormalization group. Rev Mod Phys. 2005;77(1):259–315.
41. Goreinov SA, Oseledets IV, Savostyanov DV, Tyrtyshnikov EE, Zamarashkin NL. How to find a good submatrix. In: Olshevsky V, Tyrtyshnikov E, editors. Matrix methods: theory, algorithms, applications. Hackensack, NY: World Scientific; 2010. p. 247–56.

How to cite this article: Antil H, Dolgov S, Onwunta A. TTRISK: Tensor train decomposition algorithm for risk averse optimization. Numer Linear Algebra Appl. 2023;30(3):e2481. https://doi.org/10.1002/nla.2481

APPENDIX A. CROSS APPROXIMATION

Suppose $f(\xi)$ is defined as a minimizer of a cost function $\mathcal{C}(f)$. We aim at minimizing $\mathcal{C}$ over a TT decomposition instead by driving $\nabla\mathcal{C}(\tilde{f}) = 0$. However, $\tilde{f}$ is a product expansion in the actual degrees of freedom $\{\mathbf{F}^{(k)}\}$, which makes the problem nonlinear even if $\nabla_f\mathcal{C}(f)$ was linear. Instead of running a generic nonlinear optimization of all TT cores simultaneously, we can resort to the Alternating Least Squares[38] and Density Matrix Renormalization Group[39,40] methods. Those alleviate the nonlinearity issue by iterating over $k = 1, \dots, d$, solving only $\nabla_{\mathbf{F}^{(k)}}\mathcal{C}(\tilde{f}) = 0$ in each step, similarly to the coordinate descent method. Note that $\tilde{f}$ is linear in each *individual* factor $\mathbf{F}^{(k)}$.

Applying this coordinate descent idea to the problem of interpolating a given function with a TT decomposition yields a family of *TT-Cross* algorithms.[21-23] Suppose we are given a *procedure* to evaluate a continuous function $f(\xi)$ at any given ξ. We iterate over $k = 1, \dots, d$ and in each step compute $\mathbf{F}^{(k)}$ by solving an interpolation condition

$$\tilde{f}(\xi) = f(\xi) \qquad \forall \xi \in \boldsymbol{\Xi}_k := \left\{\left(\xi_j^{(1)}, \dots, \xi_j^{(d)}\right) \in \mathbb{R}^d : j = 1, \dots, r_{k-1}n_\xi r_k\right\}, \tag{A1}$$

over some carefully chosen sampling sets $\boldsymbol{\Xi}_k$. Stretching the tensor $\mathbf{F}^{(k)}$ into a vector $f^{(k)} = [\mathbf{F}^{(k)}(s_{k-1}, i, s_k)] \in \mathbb{R}^{r_{k-1}n_\xi r_k}$, we can write (A1) as a linear equation $F_{\neq k}f^{(k)} = f(\boldsymbol{\Xi}_k)$, where each row of the matrix $F_{\neq k} \in \mathbb{R}^{r_{k-1}n_\xi r_k \times r_{k-1}n_\xi r_k}$ is given by

$$F_{\neq k}(j, s_{k-1}is_k) = \sum_{s_0,\dots,s_{k-2}} F^{(1)}_{s_0,s_1}\left(\xi_j^{(1)}\right)\cdots F^{(k-1)}_{s_{k-2},s_{k-1}}\left(\xi_j^{(k-1)}\right)\cdot \ell_i\left(\xi_j^{(k)}\right)\cdot \sum_{s_{k+1},\dots,s_d} F^{(k+1)}_{s_k,s_{k+1}}\left(\xi_j^{(k+1)}\right)\cdots F^{(d)}_{s_{d-1},s_d}\left(\xi_j^{(d)}\right), \tag{A2}$$

where $\xi_j \in \boldsymbol{\Xi}_k, j = 1, \dots, r_{k-1}n_\xi r_k$. To compute this efficiently in a recursive manner similar to (15), we restrict $\boldsymbol{\Xi}_k$ to the Cartesian form

$$\boldsymbol{\Xi}_k = \Xi_{<k} \times \Xi^{(k)} \times \Xi_{>k}, \tag{A3}$$

for some chosen sets

$$\Xi_{<k} = \left\{\left(\xi^{(1)}_{s_{k-1}}, \dots, \xi^{(k-1)}_{s_{k-1}}\right)\right\}_{s_{k-1}=1}^{r_{k-1}}, \qquad \Xi_{>k} = \left\{\left(\xi^{(k+1)}_{s_k}, \dots, \xi^{(d)}_{s_k}\right)\right\}_{s_k=1}^{r_k},$$

including $\Xi_{<1} = \Xi_{>d} = \emptyset$, and $\Xi^{(k)} = \{\xi_i^{(k)}\}$ are the quadrature nodes associated with $\{\ell_i(\xi^{(k)})\}$. This allows us to write (A2) in the form

$$F_{\neq k} = F_{<k} \otimes \ell(\Xi^{(k)}) \otimes F_{>k}, \tag{A4}$$

where $\otimes$ denotes the Kronecker product operator and

$$F_{<k} = \left[\sum_{s_0, \dots, s_{k-2}} F^{(1)}_{s_0, s_1}\left(\xi^{(1)}_j\right) \cdots F^{(k-1)}_{s_{k-2}, s_{k-1}}\left(\xi^{(k-1)}_j\right) \right] \in \mathbb{R}^{r_{k-1} \times r_{k-1}}, \qquad \left(\xi^{(1)}_j, \dots, \xi^{(k-1)}_j\right) \in \Xi_{<k}, \tag{A5}$$

$$F_{>k} = \left[\sum_{s_{k+1}, \dots s_d} F^{(k+1)}_{s_k, s_{k+1}}\left(\xi^{(k+1)}_j\right) \cdots F^{(d)}_{s_{d-1}, s_d}\left(\xi^{(d)}_j\right) \right] \in \mathbb{R}^{r_k \times r_k}, \qquad \left(\xi^{(k+1)}_j, \dots, \xi^{(d)}_j\right) \in \Xi_{>k}. \tag{A6}$$

Moreover, for the *left* and *right* sets $\Xi_{<k}, \Xi_{>k}$ we assume *nestedness* conditions

$$\Xi_{<k+1} \subset \Xi_{<k} \times \Xi^{(k)}, \qquad \Xi_{>k-1} \subset \Xi^{(k)} \times \Xi_{>k}. \tag{A7}$$

This way, given $F_{<k}$ or $F_{>k}$, we can continue the recursion by computing

$$\overline{F}_{\le k}(s_{k-1} i, s_k) = \sum_{t_{k-1}} F_{<k}(s_{k-1}, t_{k-1}) F^{(k)}_{t_{k-1}, s_k}\left(\xi^{(k)}_i\right), \quad \text{and} \tag{A8}$$

$$\overline{F}_{\ge k}(s_{k-1}, i s_k) = \sum_{t_k} F^{(k)}_{s_{k-1}, t_k}\left(\xi^{(k)}_i\right) F_{<k}(t_k, s_k), \tag{A9}$$

and extracting $F_{<k+1}$ and $F_{>k-1}$ simply as submatrices of $\overline{F}_{\le k}$ and $\overline{F}_{\ge k}$, respectively. This needs $\mathcal{O}(n_\xi r^3)$ operations.

The particular elements extracted from, for example $\Xi_{<k} \times \Xi^{(k)}$ (and $\overline{F}_{\le k}$) are chosen to ensure that $F_{<k+1}$ (and hence $F_{\ne k+1}$) are as well conditioned as possible. We use the Maximum Volume (*maxvol*) algorithm[41] which performs an iterative optimization of the volume (absolute determinant) of the submatrix $F_{<k+1} = \overline{F}_{\le k}(\mathcal{I}_{\le k}, :)$ over the index set $\mathcal{I}_{\le k} = \{s_{k-1}, i\}$ of cardinality r_k. We can apply the same algorithm to $\overline{F}^{\top}_{\ge k}$ to find an index set $\mathcal{I}_{\ge k} = \{i, s_k\}$, with cardinality $\#\mathcal{I}_{\ge k} = r_{k-1}$, that provides $|\det \overline{F}_{\ge k}(:, \mathcal{I}_{\ge k})| \approx \max_{\mathcal{I}} |\det \overline{F}_{\ge k}(:, \mathcal{I})|$. The recursion over the sampling sets (A7) is arranged by collecting $\Xi_{<k+1} = \left\{\Xi_{<k}(s_{k-1}), \xi^{(k)}_i\right\}$ for $s_{k-1}, i \in \mathcal{I}_{\le k}$, and similarly for $\Xi_{>k-1}$. The entire iteration, which we call TT-cross, is outlined in Algorithm 2.

Algorithm 2. TT-CROSS

1: Choose initial sets $\Xi_{<k}$, $k = 2, \dots, d$, stopping threshold $\delta > 0$.
2: **while** first iteration or $\|\tilde{f}(\xi) - \tilde{f}_{\text{prev}}(\xi)\| > \delta \|\tilde{f}(\xi)\|$ **do**
3: **for** $k = d, d-1, \dots, 2, 1, 2, \dots, d$ **do**
4: Sample $f(\Xi_k)$, where Ξ_k is according to (A3). ▷ $\mathcal{O}(n_\xi r^2)$ samples of f
5: Solve $F_{\ne k} f^{(k)} = f(\Xi_k)$ using the matrix (A4).
6: Compute $\mathcal{I}_{\ge k}$ from *maxvol* on $(\overline{F}_{\ge k})^{\top}$ as defined in (A9), and $\mathcal{I}_{\le k}$ from *maxvol* on $\overline{F}_{\le k}$ as defined in (A8).
7: Let $\Xi_{>k-1} = [\Xi^{(k)} \times \Xi_{>k}]|_{\mathcal{I}_{\ge k}}$, $\Xi_{<k+1} = [\Xi_{<k} \times \Xi^{(k)}]|_{\mathcal{I}_{\le k}}$, and $F_{>k-1} = \overline{F}_{\ge k}(:, \mathcal{I}_{\ge k})$, $F_{<k+1} = \overline{F}_{\le k}(\mathcal{I}_{\le k}, :)$.
8: **end for**
9: **end while**

APPENDIX B. TENSOR TRAIN ALGEBRA

Once a TT decomposition is constructed, simple operations can be performed directly with the TT cores. Besides the fast integration (15), additions of functions approximated by their TT formats, pointwise products and actions of linear operators can be written in the TT format with explicitly computable TT cores, and hence a linear complexity in d.[20] For example, addition of functions $\tilde{f}(\xi)$ and $\tilde{g}(\xi)$ defined by their TT cores $\{F^{(k)}\}$ and $\{G^{(k)}\}$ and TT ranks $(r_0, \dots, r_d)$ and $(p_0, \dots, p_d)$, respectively, is realized by the TT decomposition

$$\tilde{f}(\xi) + \tilde{g}(\xi) = \sum_{s_0, \dots, s_d = 1}^{r_0 + p_0, \dots, r_d + p_d} H^{(1)}_{s_0, s_1}(\xi^{(1)}) \cdots H^{(d)}_{s_{d-1}, s_d}(\xi^{(d)}), \quad H^{(k)}(\xi^{(k)}) = \begin{bmatrix} F^{(k)}(\xi^{(k)}) & 0 \\ 0 & G^{(k)}(\xi^{(k)}) \end{bmatrix}.$$

However, such explicit decompositions are likely to be redundant. For example, we can immediately compress $H^{(1)}$ to $[F^{(1)}(\xi^{(1)}) \quad G^{(1)}(\xi^{(1)})]$, and similarly the summation over s_d collapses $H^{(d)}$. In general, a quasi-optimal approximate *re-compression* of a TT decomposition[20] can be performed in $\mathcal{O}(dn_\xi r^3)$ operations by using truncated Singular Value decompositions (SVD) in a recursive manner similar to (15).

Linear operators acting on multivariate functions can also be decomposed (or approximated) in a similar TT format that enables a fast computation of their action. An operator $A \in \mathcal{L}(\mathcal{F}, \mathcal{F})$ with $\mathcal{F} = \mathcal{F}^{(1)} \times \cdots \times \mathcal{F}^{(d)}$ can be approximated by a TT operator $\tilde{A}$ of the form

$$\tilde{A} = \sum_{t_0,\dots,t_d=1}^{R_0,\dots,R_d} A^{(1)}_{t_0,t_1} \otimes A^{(2)}_{t_1,t_2} \otimes \cdots \otimes A^{(d)}_{t_{d-1},t_d}, \tag{B1}$$

where $A^{(k)}_{t_{k-1},t_k} \in \mathcal{L}(\mathcal{F}^{(k)}, \mathcal{F}^{(k)})$ is an operator acting on $F^{(k)}(\xi^{(k)})$. The image $\tilde{A}\tilde{f}$ can now be written as a TT decomposition

$$(\tilde{A}\tilde{f})(\xi) = \sum_{m_0,\dots,m_d=1}^{R_0 r_0,\dots,R_d r_d} B^{(1)}_{m_0,m_1}(\xi^{(1)}) \cdots B^{(d)}_{m_{d-1},m_d}(\xi^{(d)}), \qquad B^{(k)}(\xi^{(k)}) = \left[\left(A^{(k)}_{t_{k-1},t_k} F^{(k)}_{s_{k-1},s_k} \right)(\xi^{(k)}) \right],$$

where $t_k = \mathrm{mod}(m_k - 1, R_k) + 1$, and $s_k = \lfloor (m_k - 1)/R_k \rfloor + 1$. Linear equations can be solved by the Alternating Least Squares. We iterate over $k = 1, \dots, d$, solving in each step $\tilde{A}\tilde{f} = \tilde{b}$ with respect to the TT core $\mathbf{F}^{(k)}$ in the representation of $\tilde{f}$. Constructing a vector-function $f^{(\neq k)} : \mathbb{R}^d \to \mathbb{R}^{r_{k-1} n_\xi r_k}$ with elements

$$f^{(\neq k)}_{s_{k-1} i s_k}(\xi) = \sum_{s_0,\dots,s_{k-2}} F^{(1)}_{s_0,s_1}(\xi^{(1)}) \cdots F^{(k-1)}_{s_{k-2},s_{k-1}}(\xi^{(k-1)}) \cdot \ell_i(\xi^{(k)}) \cdot \sum_{s_{k+1},\dots,s_d} F^{(k+1)}_{s_k,s_{k+1}}(\xi^{(k+1)}) \cdots F^{(d)}_{s_{d-1},s_d}(\xi^{(d)}), \tag{B2}$$

similarly to (A2), we can write $\tilde{f}(\xi) = f^{(\neq k)}(\xi) f^{(k)}$, and solve a linear system

$$A_k f^{(k)} = b_k, \quad \text{where} \quad A_k = \left[\langle f^{(\neq k)}_s, \tilde{A} f^{(\neq k)}_t \rangle \right], \quad b_k = \left[\langle f^{(\neq k)}_s, \tilde{b} \rangle \right],$$

in each step. If $\tilde{A}$ and $\tilde{b}$ are given by TT decompositions, A_k and b_k can be computed recurrently in the course of iteration using matrix products similar to (A8), (A9). One full iteration over $k = 1, \dots, d$ needs $\mathcal{O}(dn^2_\xi R^2 r^2 + dn_\xi R r^3)$ operations,[38] where $R := \max_k R_k$.