Weierstraß-Institut
für Angewandte Analysis und Stochastik

Leibniz-Institut im Forschungsverbund Berlin e. V.

Preprint ISSN 2198-5855

Generative modelling with tensor train approximations of Hamilton–Jacobi–Bellman equations

David Sommer[1], Robert Gruhlke[2], Max Kirstein[3], Martin Eigel[1], Claudia Schillings[2]

submitted: December 21, 2023

[1] Weierstrass Institute
Mohrenstr. 39
10117 Berlin
Germany
E-Mail: david.sommer@wias-berlin.de
martin.eigel@wias-berlin.de

[2] FU Berlin
Animallee 6
14195 Berlin
Germany
E-Mail: r.gruhlke@fu-berlin.de
c.schillings@fu-berlin.de

[3] Bosch Center for
Artificial Intelligence
31102 Hildesheim
Germany
E-Mail: max.kirstein@de.bosch.com

No. 3078
Berlin 2023

2020 *Mathematics Subject Classification.* 35F21, 35Q84, 62F15, 65N75, 65C30,

Key words and phrases. Generative modelling, approximate sampling, Hamilton-Jacobi-Bellman, low rank tensors.

DS & ME acknowledge support by the Profit project *ReLkat - Reinforcement Learning for complex automation engineering* as well as support by the ANR-DFG project *COFNET: Compositional functions networks - adaptive learning for high-dimensional approximation and uncertainty quantification*. RG, ME & CS acknowledge support by the DFG MATH+ project AA5-5 (was EF1-25) - *Wasserstein Gradient Flows for Generalised Transport in Bayesian Inversion*. ME acknowledges partial funding by the DFG priority program SPP 2298 “Theoretical Foundations of Deep Learning”. This study does not have any conflicts to disclose.

Edited by
Weierstraß-Institut für Angewandte Analysis und Stochastik (WIAS)
Leibniz-Institut im Forschungsverbund Berlin e. V.
Mohrenstraße 39
10117 Berlin
Germany

Fax: +49 30 20372-303
E-Mail: preprint@wias-berlin.de
World Wide Web: http://www.wias-berlin.de/

Generative modelling with tensor train approximations of Hamilton–Jacobi–Bellman equations

David Sommer, Robert Gruhlke, Max Kirstein, Martin Eigel , Claudia Schillings

Abstract

Sampling from probability densities is a common challenge in fields such as Uncertainty Quantification (UQ) and Generative Modelling (GM). In GM in particular, the use of reverse-time diffusion processes depending on the log-densities of Ornstein-Uhlenbeck forward processes are a popular sampling tool. In [5] the authors point out that these log-densities can be obtained by solution of a *Hamilton-Jacobi-Bellman* (HJB) equation known from stochastic optimal control. While this HJB equation is usually treated with indirect methods such as policy iteration and unsupervised training of black-box architectures like Neural Networks, we propose instead to solve the HJB equation by direct time integration, using compressed polynomials represented in the Tensor Train (TT) format for spatial discretization. Crucially, this method is sample-free, agnostic to normalization constants and can avoid the curse of dimensionality due to the TT compression. We provide a complete derivation of the HJB equation's action on Tensor Train polynomials and demonstrate the performance of the proposed time-step-, rank- and degree-adaptive integration method on a nonlinear sampling task in 20 dimensions.

1 Introduction and related work

Consider the problem of sampling from a probability measure μ_* on $\mathbb{R}^d$, $d \in \mathbb{N}$, with Lebesgue-density

$$\pi_*(y) = \frac{1}{Z} \exp(-\Phi(y)), \tag{1.1}$$

where $\Phi \colon \mathbb{R}^d \to \mathbb{R}$ is a sufficiently regular function called the *potential* and $Z \in (0, \infty)$ is a normalization constant such that $\int_{\mathbb{R}^d} \pi_*(y) \mathrm{d}y = 1$. Throughout this manuscript, we assume that Φ is known and can be evaluated, while the normalization constant Z is unknown and difficult or even impossible to compute. Over time, a myriad of different sampling methods have been devised, including Markov Chain Monte Carlo (MCMC) methods [35, 6, 34], methods based on Stein variational gradient descent [24], Langevin dynamics [36, 14, 15, 32, 7, 12], or Langevin dynamics preconditioned with measure transport [41] to name just a few. In the last few years, interacting particle systems have received a lot of attention [16, 14, 15, 32, 7, 12]. An important application where one aims to sample from densities of the form (1.1) stems from solutions of inverse problems via Bayesian inference [40].

Since our approach is linked to (interacting particle-) Langevin samplers, we take a moment to review these methods in more detail. All methods proposed in [14, 15, 32, 7] work with an Itô diffusion process of the form

$$\mathrm{d}X_t = f(X_t)\mathrm{d}t + g(X_t)\mathrm{d}W_t, \tag{1.2}$$

where W is a standard Brownian motion with appropriate dimension, f is the drift and g is the diffusion. Under certain assumptions on the potential, e.g. (strong) convexity, convexity at infinity, regularity

DOI 10.20347/WIAS.PREPRINT.3078 Berlin 2023

and growth conditions, this process is ergodic [42, 15] and admits either μ_* (in the case of a single particle process) or $\otimes_{i=1}^{B}\mu_*$ (in the case of a system of $B \in \mathbb{N}$ interacting particles) as an invariant measure.

Samples from μ_* are obtained by propagating an initial batch of arbitrarily distributed samples through the process (1.2) for infinite time. In the classical overdamped Langevin dynamics, the drift term f is given by the negative gradient $-\nabla\Phi$ of the potential. In state-of-the-art interacting particle methods like the *Affine Invariant Langevin Dynamics* (ALDI) [14], this drift is modified by a reversible perturbation of the underlying process (see e.g. [41, Equation 2.4] for a general definition of reversible perturbations and [15, Definition 3.1] for the specific perturbation of ALDI). Reversible perturbations can increase convergence speed [33], while ensuring that the perturbed SDE maintains the same invariant measure as the unperturbed system and is still time-reversible. Even if the resulting system is time-reversible, the reverse-time process is not considered in those works, since the *forward* process (1.2) admits μ_* as invariant measure.

While the time-homogeneous drift term of (1.2) makes these methods conceptually simple, it comes with a potential downside with regard to the class of measures μ_* that can be approximated. ALDI comes with theoretical convergence guarantees only in the case of a potential with Gaussian tails outside of a compact set [14]. In [12], the authors propose using a time-inhomogeneous process

$$\mathrm{d}X_t = f(t, X_t)\mathrm{d}t + g(X_t)\mathrm{d}W_t, \tag{1.3}$$

where $f(t,\cdot)$ is defined by gradients of log-densities of intermediate measures defined upon time dependent interpolation, e.g. a convex combination of the target potential Φ and a simpler auxiliary potential. By the choice of the auxiliary measure, the flow towards the target distribution is fixed. While this so called *homotopy*-approach can substantially increase convergence speed in practice, e.g. to sample from multimodal target distributions, the choice of auxiliary measures allowing for optimal flows remains an open question.

Contrary, reverse-time diffusion processes offer a principled way of defining a process of the form (1.3), which can be used to sample from μ_*. The key observation, dating back to [1], is that the reverse-time process corresponding to (1.3) defines again a diffusion process of the form (1.3). For some years, this property has been used in what is now called *Diffusion Generative Modelling* [37, 18, 39]. In contrast to Bayesian inference, where μ_* is known but difficult to sample from, the goal here is to generate new samples from some completely unknown data distribution of which a finite number of samples $\{x_i\}_{i=1}^{D}$, $D \in \mathbb{N}$, are available. The central idea is to use an Ornstein-Uhlenbeck process mapping any distribution to a standard-normal distribution $\mathcal{N}(0, I_d)$ for $t \to \infty$ and then, by using the available samples $\{x_i\}_{i=1}^{D}$, learning the drift of the reverse-time process, mapping $\mathcal{N}(0, I_d)$ back to the data distribution [39]. More specifically, the gradient-log-density or *score* of the Ornstein-Uhlenbeck process is learned by minimizing a *score-matching* objective function [22, 38], which is essentially a weighted time-average of mean-squared errors (see e.g. [39, Equation 7]). The score determines the reverse process. Once the score is known, new samples from the data distribution can be obtained by sampling from the standard-normal distribution and propagating the samples through the reverse process. However, classical score-matching relies on the samples $\{x_i\}_{i=1}^{D}$ of the data distribution, which are usually not available in a Bayesian setting. Hence, we consider an alternative approach.

The authors of [5] point out that the negative log-density of a reverse-time diffusion process satisfies a *Hamilton-Jacobi-Bellman* (HJB) equation. Since the score is invariant under additive constants to the log-density, it suffices to solve this HJB equation up to an additive constant to obtain the correct score. In particular, the normalization constant of the target density need not be known. Hence, solving the corresponding HJB equation is a viable method of obtaining the score in a Bayesian setting.

Tensor Trains [29] have been used successfully in several works on approximations of HJB equations for nonlinear optimal control, see e.g. [31, 11] and references therein. In [31] the solution of the deterministic finite horizon HJB equation is obtained by a combination of Monte-Carlo (MC) sampling and policy iteration. While this approach is appealing due to its model-free nature, the policy iteration requires the solution of multiple nonlinear optimization problems at each time step. Furthermore, MC sampling may lead to slow convergence. In [11] a spectral discretization is used, circumventing the slow convergence rate of MC sampling and achieving algebraic convergence for a class of deterministic infinite horizon optimal control problems. In contrast to these works, we propose a method not reliant on policy iteration. Furthermore, no nonlinear optimization has to be performed except at the initial time point. Instead, the HJB right-hand side is discretized by orthogonal projection onto polynomial space, resulting in an ODE in tensor space. Subsequently, this ODE is integrated using methods for time-integration of Tensor Trains.

1.1 Contribution and Outline

The main contribution of this work lies in providing an interpretable solver based on compressed polynomials for the reverse-time HJB equation as it appears in the context of Generative Modelling and Bayesian Inference. Specifically, we integrate the HJB equation using orthogonal projections of the right-hand side and rank-retractions onto a smooth manifold within polynomial space defined by Tensor Trains of a fixed rank. The solver adaptively chooses its stepsize based on current projection- and retraction-errors as well as the local stiffness, which is estimated by local linearizations of the HJB. This approach is sample-free and agnostic to normalization constants and can therefore be used in a Bayesian setting. We demonstrate the performance of the solver on a nonlinear test case in $d = 20$ dimensions.

The outline of the rest of the paper is as follows.

- Section 2 covers the relevant theory of diffusion processes necessary to construct a process of the form (1.3), which can be used to sample from μ_*. In particular, Remark 2.1 offers one such form as a reverse-time Ornstein-Uhlenbek process. The corresponding reverse-time HJB equation determining the score of this process is given in (2.5).

- In Section 3 we introduce our approximation class for the log-densities, namely functional Tensor Trains with orthogonal polynomial ansatz functions. A motivation for this ansatz class can be found in Appendix C. This section further introduces all algebraic operations on tensor space necessary to solve a projected version of the HJB equation.

- Section 4 is the main part of the paper, where we are concerned with the solution of the HJB. We state the equivalence of the HJB projected onto polynomial space of fixed degree with an ODE in tensor space (Theorem 4.1). Furthermore, we give a precise version of the proposed solution algorithm (Algorithm 2).

- Finally, the performance of the solver is demonstrated on a Gaussian test case as well as a 20-dimensional nonlinear potential in Section 5.

2 Reverse-time diffusion processes and HJB equation

Let the terminal time $T > 0$ and a d-dimensional Ornstein-Uhlenbeck process $(X_t)_{t\in[0,T]}$ be defined by

$$\mathrm{d}X_t = -X_t\mathrm{d}t + \sqrt{2}\mathrm{d}W_t, \qquad X_0 \sim \mu_*, \tag{2.1}$$

where W_t denotes standard d-dimensional Brownian motion. The probability density function π_t of this process satisfies the Fokker-Planck equation

$$\partial_t \pi_t = \Delta\pi_t + x \cdot \nabla\pi_t + d\pi_t, \quad \pi_0 = \pi_*, \tag{2.2}$$

for $t \in [0, T]$. Since the (standard normal) invariant measure of (2.1) satisfies a log-Soboloev inequality, the corresponding law μ_{X_t} of (2.1) converges exponentially in Kullback-Leibler divergence (KL) to the standard normal distribution $\mathcal{N}(0, I_d)$ on $\mathbb{R}^d$ [27], i.e.

$$\mathrm{KL}(\mu_{X_t}||\mathcal{N}(0, I_d)) \leq e^{-2t}\mathrm{KL}(\mu_*||\mathcal{N}(0, I_d)). \tag{2.3}$$

Hence, for sufficiently large T, the measure μ_{X_T} will be close to a standard normal distribution in KL divergence. The following remark provides a reverse-time process $(Y_t)_{t\in[0,T]}$ with $Y_0 \sim \mu_{X_T}$ and $Y_T \sim \mu_*$.

Remark 2.1 (Reverse-time Ornstein-Uhlenbeck process)**.** *Let $(X_t)_{t\in[0,T]}$ be defined by (2.1). Then, for any $\lambda \in [0, 1]$ the process $(Y_t)_{t\in[0,T]}$ defined by*

$$\mathrm{d}Y_t = [Y_t + (2 - \lambda)\nabla \log \pi_{T-t}(Y_t)]\,\mathrm{d}t + \sqrt{2(1 - \lambda)}\mathrm{d}W_t, \qquad Y_0 \sim \mu_{X_T} \tag{2.4}$$

satisfies $\mu_{Y_t} = \mu_{X_{T-t}}$ and in particular $\mu_{Y_T} = \mu_*$. This result is an immediate consequence of [21, Appendix G], which covers a much wider range of diffusion processes. The most common choices for λ are $\lambda = 0$, used for the reverse process e.g. in [39], and $\lambda = 1$, which leads to a reverse ODE known as probability flow ODE [39].

To formulate the reverse process $(Y_t)_{t\in[0,T]}$ we need the score $\nabla \log \pi_t$. If a sufficient number of samples of μ_* are available, we can apply score matching techniques (see [37, 39] and references therein). Lacking these samples, we could try to obtain π_t by solving (2.2), but the fact that π_t needs to be a density for every t makes this approach cumbersome for approximation methods. Instead, we apply a *Hopf-Cole transformation* $v_t := -\log \pi_t$ to (2.2). A short calculation by product and chain rule (see Appendix A) yields that v_t satisfies the PDE

$$\partial_t v_t = \Delta v_t + x \cdot \nabla v_t - \|\nabla v_t\|_2^2 - d, \qquad v_0 = -\log \pi_*, \tag{2.5}$$

for $t \in [0, T]$. This nonlinear PDE is the time-reverse of a HJB equation appearing in finite-horizon stochastic optimal control. As [5] pointed out, we can now apply techniques from optimal control to approximate the score. A straightforward way is to approximately solve the HJB equation (2.5) with some suitable class of functions such as Neural Networks [44, 4, 28]. Instead of this black-box approach, we propose solving (2.5) by means of compressed polynomials represented by a low-rank tensor format, the details of which are provided in the next section. In contrast to Neural Networks, this approach is highly interpretable and utilizes the structure of the HJB equation. In particular, we make frequent use

of the fact that the right-hand side $F(v) := \Delta v + x \cdot \nabla v - \|\nabla v\|_2^2 - d$ of (2.5) can be split into a constant, linear and nonlinear contribution, given by

$$\mathrm{Const}(v) = d, \tag{2.6}$$
$$\mathrm{Lin}(v) = \Delta v + x \cdot \nabla v, \tag{2.7}$$
$$\mathrm{NonLin}(v) = -\|\nabla v\|^2. \tag{2.8}$$

Before going into the detail about the polynomial approximation in the following section, we briefly sketch some of the core ideas.

First, we note the constant term (2.6) can be dropped from (2.5) since the score is agnostic to constant shifts of the log-density. More precisely, v_t is a solution to (2.5) if and only if $\overline{v}_t := v_t + td$ is a solution to $\partial_t \overline{v}_t = \Delta \overline{v}_t + x \cdot \nabla \overline{v}_t - \|\nabla \overline{v}_t\|_2^2, \overline{v}_0 = -\log \pi_*, t \in [0, T]$. The two solutions v_t and $\overline{v}_t$ differ only by a constant shift for every $t \in [0, t]$, hence the score satisfies $\nabla \log \pi_t = -\nabla v_t = -\nabla \overline{v}_t$. By the same reasoning, an arbitrary constant can be added to the initial condition of (2.5) without affecting the score. By choosing this constant equal to $-\log(Z)$, we achieve $-\log \pi_* - \log(Z) = \Phi$. Thus, from now on we consider the equation

$$\partial_t v_t = \mathrm{Lin}(v_t) + \mathrm{NonLin}(v_t), \qquad v_0 = \Phi. \tag{2.9}$$

Morevover, if v_t is a polynomial of fixed degree $N \in \mathbb{N}$ for any t, then $\mathrm{Lin}(v_t)$ is also a polynomial of degree N. This means that if v_0 is a polynomial of a fixed degree, integrating only the linear part of (2.9) would yield a polynomial of same degree for all $t \in [0, T]$. For the nonlinear part $\mathrm{NonLin}(v_t)$ this is only true for $N = 2$. In this quadratic case, (2.9) can be solved to arbitrary accuracy. In particular, if μ_* is a zero mean Gaussian with density

$$\pi_*(x) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{x^\intercal \Sigma^{-1} x} \tag{2.10}$$

for positive definite $\Sigma \in \mathbb{R}^{d,d}$, then (2.9) corresponds to the HJB equation of a linear-quadratic optimal control problem with solution given by $v_t(x) = x^\intercal P_t x$, where $P_t \in \mathbb{R}^{d,d}$, $t \in [0, T]$, solves a Ricatti matrix differential equation (see Appendix B).

Solving (2.9) e.g. with an explicit Euler method for time discretization leads to a steady increase of the degree over time for all initial degrees larger than $N = 2$. This is due to the nonlinear term: if v_t for some $t \in [0, T]$ is a polynomial of degree N, then $\mathrm{NonLin}(v_t)$ is (in general) a polynomial of degree $\leq 2N$. To prevent this degree increase, we *project* the nonlinear part of the right-hand side back onto the space spanned by polynomials of degree N before performing the time integration step. Furthermore, since the linear space of polynomials suffers from the curse of dimensionality, we use a *compression* or *retraction* after every time step, finding a best approximation of the new iterate in a low dimensional manifold. In the case of an explicit Euler method, the resulting integration scheme can be written as

$$v_{t+\tau_t} = \mathrm{Compression}\left[v_t + \tau_t \left(\mathrm{Lin}(v_t) + \mathrm{Projection}\left[\mathrm{NonLin}(v_t)\right]\right)\right], \tag{2.11}$$

where $\tau_t > 0$ is the current adaptively chosen stepsize. The precise definition of all terms involved is the subject of the next section.

3 Functional Tensor Trains (FTT) and Tensor Trains (TT)

In this section we introduce the approximation class used as a spatial discretization for the HJB equation. Let $K \subset \mathbb{R}^d$ be a compact hypercube defined by $a_i, b_i \in \mathbb{R}$ with $a_i < b_i$ for $i = 1, \ldots, d$

$\boldsymbol{n} \in \mathbb{N}_0^d$	dimension array $\boldsymbol{n} = (n_1, \dots, n_d)$
$k\boldsymbol{n} + l$	$(kn_1 + l, \dots, kn_d + l)$ for $k, l \in \mathbb{N}_0$
$[\boldsymbol{n}]$	indexing $[\boldsymbol{n}] = \bigtimes_{i=1}^d \{0, \dots, n_i\}$
$\boldsymbol{n}_1 \geq \boldsymbol{n}_2, \boldsymbol{n} \geq k$	component wise comparison $\boldsymbol{n}, \boldsymbol{n}_1, \boldsymbol{n}_2 \in \mathbb{N}^d$, $k \in \mathbb{N}$
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$	multiindex in $\mathbb{N}_0^d$, note that we always index starting from 0
$\mathbb{R}^{\boldsymbol{n}}$	tensor space $\mathbb{R}^{n_1, \dots, n_d}$
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	tensor elements in $\mathbb{R}^{\boldsymbol{n}}$
$\boldsymbol{r}$	rank $\boldsymbol{r} = (r_1, \dots, r_{d-1})$ in $\mathbb{N}^{d-1}$
$\boldsymbol{r}^1 \boldsymbol{r}^2$	multiplication $\boldsymbol{r}^1 \boldsymbol{r}^2 = (r_1^1 r_1^2, \dots, r_{d-1}^1 r_{d-1}^2)$ in $\mathbb{N}^{d-1}$
k_i, l_i	rank enumeration indices in $\{1, \dots, r_i\}$
A_i, B_i, C_i	component order 3 tensor in $\mathbb{R}^{r_{i-1}, n_i+1, r_i}$ with entries indexed by $[k_{i-1}, \alpha_i, k_i]$
$A_i[\alpha_i]$	matrix extraction $A_i[\alpha_i] = A_i[:, \alpha_i, :] \in \mathbb{R}^{r_{i-1}, r_i}$ of component tensor A_i
$A_i[k_{i-1}, :, k_i]$	vector extraction in $\mathbb{R}^{n_i+1}$ for each rank enumeration k_{i-1}, k_i
$\boldsymbol{A}[\boldsymbol{\alpha}]$	tensor indexing $\boldsymbol{A}[\alpha_1, \dots, \alpha_d]$ for $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{n}}$, $\boldsymbol{\alpha} \in [\boldsymbol{n}]$, $\boldsymbol{n} \in \mathbb{N}_0^d$

Table 1: List of compact notations used in this work.

and $K = \bigtimes_{i=1}^{d}[a_i, b_i]$. A function $f\colon K \to \mathbb{R}$ is said to have functional Tensor Train (FTT) [30] rank $\overline{\boldsymbol{r}} = (\overline{r}_1, \dots, \overline{r}_{d-1}) \in \mathbb{N}^{d-1}$ with the convention $\overline{r}_0 = \overline{r}_d = 1$, if it can be written as

$$f(x_1, \dots, x_d) = F_1(x_1)F_2(x_2)\cdots F_d(x_d) \tag{3.1}$$

with matrix valued functions $F_i(x_i) \in \mathbb{R}^{\overline{r}_{i-1}, \overline{r}_i}$, $x_i \in [a_i, b_i]$ for $i = 1, \dots, d$. For discussions regarding the approximation of functions of mixed regularity or compositional structures we refer to [3, 17, 2].

In order to obtain a discrete approximation class, for each $i = 1, \dots, d$ and $\alpha \in \mathbb{N}_0$ let p_α^i denote the α-th orthonormal Legendre polynomial with respect to the standard L^2 inner product on $[a_i, b_i]$. For $\boldsymbol{n} \in \mathbb{N}_0^d$, we define the discrete set of orthonormal polynomials of degree $\boldsymbol{n}$ by

$$\Pi_{\boldsymbol{n}} \coloneqq \{p_{\boldsymbol{\alpha}} \coloneqq \bigotimes_{i=1}^{d} p_{\alpha_i}^i \mid \boldsymbol{\alpha} \in [\boldsymbol{n}]\}, \tag{3.2}$$

where $[\boldsymbol{n}]$ is defined as in Table 1. For f with FTT rank $\overline{\boldsymbol{r}}$, we then may approximate

$$f(x_1, \dots, x_d) \approx \sum_{\boldsymbol{\alpha} \in [\boldsymbol{n}]} \boldsymbol{C}[\boldsymbol{\alpha}] p_{\boldsymbol{\alpha}}(x_1, \dots, x_d), \tag{3.3}$$

with a tensor array $\boldsymbol{C} \in \mathbb{R}^{\boldsymbol{n}+1}$ with Tensor Train (TT) rank $\boldsymbol{r} = (r_1, \dots, r_{d-1})^\intercal \in \mathbb{N}^{d-1}$ bounded by the FTT rank $\overline{\boldsymbol{r}}$. In particular we have the decomposition into a Tensor Train (or Matrix Product State) format

$$\boldsymbol{C}[\boldsymbol{\alpha}] = C_1[\alpha_1]C_2[\alpha_2]\cdots C_d[\alpha_d], \tag{3.4}$$

with matrices $C_i[\alpha_i] \in \mathbb{R}^{r_{i-1}, r_i}$ and the convention that $r_0 = r_d = 1$. Note that the relation of $\overline{\boldsymbol{r}}$ and $\boldsymbol{r}$ depends on the relation of F_i and the polynomials in i-th direction. In particular it holds $\overline{\boldsymbol{r}} = \boldsymbol{r}$ if for all $i = 1, \dots, d$ and $\alpha = 0, \dots, n_i$ it holds

$$\int_{a_i}^{b_i} F_i(x_i) p_\alpha^i(x_i) \mathrm{d}x_i \neq \mathbf{0} \in \mathbb{R}^{\overline{r}_{i-1}, \overline{r}_i}.$$

Provided that the ranks can be bounded, the TT format exhibits a storage complexity of $\mathcal{O}(\max(n_1, \dots, n_d) d \max(r_1, \dots, r_{d-1})^2)$, which scales only linearly in the dimension d, hence avoiding the curse of dimensionality. The set of such Tensor Trains of fixed rank $\boldsymbol{r}$ defines a manifold $\mathcal{M}_{\boldsymbol{r}} \subset \mathbb{R}^{\boldsymbol{n}+1}$, see e.g. [20].

As a first step of our HJB solver, we propose to approximate $V_0 = -\log \pi^*$ in a functional Tensor Train format based on orthogonal polynomial space discretization as in (3.3) for some TT rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$. A motivation for this type of approximation for Bayesian posteriors can be found in Appendix C. In what follows we discuss the actions of the linear and nonlinear operators defined in (2.7) and (2.8) on functions given in that format. To that end, we define for any tensor $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{n}+1}$ the associated polynomial $v_A \in \operatorname{span} \Pi_{\mathbf{n}}$ by

$$v_{\boldsymbol{A}} = \sum_{\boldsymbol{\alpha} \in [\boldsymbol{n}]} \boldsymbol{A}[\boldsymbol{\alpha}] p_{\boldsymbol{\alpha}}. \tag{3.5}$$

3.1 The linear part

This section is concerned with the operator Lin from (2.7), appearing in the right-hand side of the HJB in (2.9).

Let the differential operator $\mathcal{D}\colon \mathcal{C}^2(\mathbb{R}) \to \mathcal{C}(\mathbb{R})$ be defined as $\mathcal{D}v = \partial_x^2 v + x\partial_x v$ for $v \in \mathcal{C}^2(\mathbb{R})$ and let $\mathcal{I}\colon \mathcal{C}^2(\mathbb{R}) \to \mathcal{C}^2(\mathbb{R})$ denote the identity operator. Then, it holds

$$\mathrm{Lin} = \mathcal{D}\otimes\mathcal{I}\otimes\ldots\otimes\mathcal{I} + \mathcal{I}\otimes\mathcal{D}\otimes\mathcal{I}\otimes\ldots\otimes\mathcal{I} + \ldots + \mathcal{I}\otimes\ldots\otimes\mathcal{I}\otimes\mathcal{D}. \tag{3.6}$$

As a first result we discuss the effect of the operator on functions v given in FTT format.

Lemma 3.1. *Let $f \in \mathcal{C}^2(K)$ have FTT-rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$. Then, $\mathrm{Lin}(f)$ has FTT-rank at most $2\boldsymbol{r}$.*

Proof. The assertion follows immediately since $\mathrm{Lin}(f)$ defines a *Laplace-like sum* of FTTs, meaning that each summand only modifies a single component of the FTT. More precisely, we have

$$\begin{aligned}\mathrm{Lin}(f)(x) = \begin{bmatrix}\mathcal{D}F_1(x_1) & F_1(x_1)\end{bmatrix}\begin{bmatrix}F_2(x_2) & 0\\ \mathcal{D}F_2(x_2) & F_2(x_2)\end{bmatrix}\cdots\\ \cdots\begin{bmatrix}F_{d-1}(x_{d-1}) & 0\\ \mathcal{D}F_{d-1}(x_{d-1}) & F_{d-1}(x_{d-1})\end{bmatrix}\begin{bmatrix}F_d(x_d)\\ \mathcal{D}F_d(x_d)\end{bmatrix},\end{aligned} \tag{3.7}$$

which defines a product of matrix valued functions as in (3.1). The rank bound follows immediately from the block structure of (3.7) and the dimensions of $F_i, \mathcal{D}F_i$ for $i = 1,\ldots,d$. □

When applied to polynomials, the linear operator can be expressed in terms of its action on the polynomial's coefficients. More precisely, the discretization of Lin on the finite set $\Pi_{\boldsymbol{n}}$ for some $\boldsymbol{n} = (n_1,\ldots,n_d)^\intercal \in \mathbb{N}_0^d$ implies a linear operator $\boldsymbol{L}\colon \mathbb{R}^{\boldsymbol{n}+1} \to \mathbb{R}^{\boldsymbol{n}+1}$ given as

$$\boldsymbol{L} := \sum_{i=1}^{d} \boldsymbol{L}_i, \quad \boldsymbol{L}_i := \left(\bigotimes_{j=1}^{i-1} I_{n_j+1}\right)\otimes D_i \otimes \left(\bigotimes_{j=i+1}^{d} I_{n_j+1}\right), \tag{3.8}$$

with identity matrix $I_n \in \mathbb{R}^{n,n}$. For the structure of the matrix $D_i \in \mathbb{R}^{n_i+1,n_i+1}$ we refer to Appendix E.1, specifically equation (E.5). For the moment it suffices to note that D_i governs the action of the differential operator $\mathcal{D}$ on the coefficients of the polynomials in dimension i. It can be shown that the action of Lin on a polynomial corresponds to algebraic manipulation of the coefficient tensor with respect to $\boldsymbol{L}$, which is the result of the following lemma.

Lemma 3.2. *Let $\boldsymbol{n} \in \mathbb{N}_0^d$, Lin and $\boldsymbol{L}$ from (3.6) and (3.8). Then, for $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{n}+1}$ we have*

$$\mathrm{Lin}\, v_{\boldsymbol{A}} = v_{\boldsymbol{L}\boldsymbol{A}}. \tag{3.9}$$

Proof. Let $\boldsymbol{L}_i[\boldsymbol{\beta},\boldsymbol{\alpha}] := \left(\bigotimes_{j=1}^{i-1} I_{n_j+1}[\beta_j,\alpha_j]\right)\otimes D_{n_i}[\beta_i,\alpha_i]\otimes\left(\bigotimes_{j=i+1}^{d} I_{n_j+1}[\beta_j,\alpha_j]\right)$. Then, the action of $\boldsymbol{L}_i$ on $\boldsymbol{A}$ defines a tensor $\boldsymbol{B}_i$ given as $\boldsymbol{B}_i[\boldsymbol{\beta}] = \sum_{\boldsymbol{\alpha}\in[\boldsymbol{n}]} \boldsymbol{L}_i[\boldsymbol{\beta},\boldsymbol{\alpha}]\boldsymbol{A}[\boldsymbol{\alpha}]$. Moreover,

$\boldsymbol{LA} = \sum_{i=1}^{d} \boldsymbol{B}_i$. Hence,

$$\begin{aligned}
\operatorname{Lin} v_{\boldsymbol{A}} &= \sum_{i=1}^{d} \sum_{\boldsymbol{\alpha}\in[\boldsymbol{n}]} \left(\bigotimes_{j=1}^{i-1} \mathcal{I}\right) \otimes \mathcal{D} \otimes \left(\bigotimes_{j=i+1}^{d} \mathcal{I}\right) \boldsymbol{A}[\boldsymbol{\alpha}] p^1_{\alpha_1} \otimes \cdots \otimes p^d_{\alpha_d} \\
&= \sum_{i=1}^{d} \sum_{\boldsymbol{\beta}\in[\boldsymbol{n}]} \sum_{\boldsymbol{\alpha}\in[\boldsymbol{n}]} \boldsymbol{L}_i[\boldsymbol{\beta},\boldsymbol{\alpha}] \boldsymbol{A}[\boldsymbol{\alpha}] p^1_{\alpha_1} \otimes \cdots \otimes p^d_{\alpha_d} \\
&= \sum_{i=1}^{d} \sum_{\beta\in[\boldsymbol{n}]} \boldsymbol{B}_i[\boldsymbol{\beta}] p^1_{\beta_1} \otimes \cdots \otimes p^d_{\beta_d} \\
&= v_{\sum_{i=1}^{d} \boldsymbol{B}_i}
\end{aligned}$$

□

The contraction $\boldsymbol{LA}$ is cumbersome for full tensors $\boldsymbol{A}$. However, it is easy to implement if $\boldsymbol{A} \in \mathcal{M}_{\boldsymbol{r}}$ is a Tensor Train of fixed rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$, such that $\boldsymbol{A}[\alpha] = A_1[\alpha_1]A_2[\alpha_2]\cdots A_d[\alpha_d]$ with $A_i[\alpha_i] \in \mathbb{R}^{r_{i-1},r_i}$ for $i = 1,\dots,d$. In this case, let $D_{i,A_i}[\beta_i] := \sum_{\alpha_i=0}^{n_i} D_i[\beta_i,\alpha_i]A_i[\alpha_i]$ for $i = 1,\dots,d$. Then,

$$\begin{aligned}
(\boldsymbol{LA})[\beta] &= \sum_{i=1}^{d} \sum_{\alpha\in[\boldsymbol{n}]} \boldsymbol{L}_i[\beta,\alpha] A[\alpha] \\
&= \sum_{i=1}^{d} \left(\bigotimes_{j=1}^{i-1} A_j[\beta_j]\right) \otimes D_{i,A_i}[\beta_i] \otimes \left(\bigotimes_{j=i+1}^{d} A_j[\beta_j]\right) \\
&= \begin{bmatrix} D_{1,A_1}[\beta_1]^\intercal & A_1[\beta_1]^\intercal \end{bmatrix} \begin{bmatrix} A_2[\beta_2]^\intercal & 0 \\ D_{2,A_2}[\beta_2]^\intercal & A_2[\beta_2]^\intercal \end{bmatrix} \cdots \\
&\quad \cdots \begin{bmatrix} A_{d-1}[\beta_{d-1}]^\intercal & 0 \\ D_{d-1,A_{d-1}}[\beta_{d-1}]^\intercal & A_{d-1}[\beta_{d-1}]^\intercal \end{bmatrix} \begin{bmatrix} A_d[\beta_d] \\ D_{d,A_d}[\beta_d] \end{bmatrix}.
\end{aligned}$$

Hence, $\boldsymbol{LA}$ is given in TT format through a *Laplace-like sum* with TT rank bounded by $2\boldsymbol{r}$, which is consistent with Lemma 3.1. In particular, this formula involves only contractions of the matrices $D_i \in \mathbb{R}^{n_i+1,n_i+1}$ with the order three tensors $A_i \in \mathbb{R}^{r_{i-1},n_i+1,r_i}$ for $i = 1,\dots,d$. No contraction with the full tensor $\boldsymbol{A}$ is required.

3.2 The nonlinear part

This section is concerned with the operator $\operatorname{NonLin}(\,\cdot\,) = \|\nabla\,\cdot\,\|^2$ from (2.8), appearing in the right-hand side of the HJB equation in (2.9), in case the arguments are functions given in FTT format. This operator is a combination of partial derivatives, squares and a summation. We split the results into two Lemmas. First, we derive a more general bound on the FTT-rank of a product of functions with bounded FTT-rank.

Lemma 3.3. *Let g, f have FTT-rank $\boldsymbol{r}^f$ and $\boldsymbol{r}^g$, respectively. Then $g \cdot f$ has FTT-rank at most $\boldsymbol{r}^g\boldsymbol{r}^f$.*

Proof. We write $f(x) = F_1(x_1)\cdot\ldots\cdot F_d(x_d)$ and $g(x) = G_1(x_1)\cdot\ldots\cdot G_d(x_d)$ with $F_i(x_i) \in \mathbb{R}^{r^f_{i-1},r^f_i}$, $G_i(x_i) \in \mathbb{R}^{r^g_{i-1},r^g_i}$ for $i = 1,\ldots,d$. Let $\otimes_k$ denote the standard matrix-Kronecker product with the convention that for two scalar values $a, b \in \mathbb{R}$ we set $a \otimes_k b = a \cdot b$. Then, we have

$$f(x)g(x) = F_1(x_1)\cdot\ldots\cdot F_d(x_d)\cdot G_1(x_1)\cdot\ldots\cdot G_d(x_d) = F_1(x_1)\otimes_k G_1(x_1)\cdot\ldots\cdot F_d(x_d)\otimes_k G_d(x_d),$$

where $F_i(x_i) \otimes_k G_i(x_i) \in \mathbb{R}^{r^f_{i-1}r^g_{i-1},r^f_i r^g_i}$. □

Second, the rank bound of the nonlinear right hand side is provided.

Lemma 3.4. *Let f have FTT-rank $\boldsymbol{r}$, then $\mathrm{NonLin}(f) = \|\nabla f\|^2 = \sum_{i=1}^d \left(\frac{\partial f}{\partial x_i}\right)^2$ has FTT-rank at most $2\boldsymbol{r}^2$.*

Proof. Note that $\frac{\partial f}{\partial x_i} = F_1(x_1)\ldots\partial_{x_i}F_i(x_i)\ldots F_d(x_d)$ has FTT-rank $\leq \boldsymbol{r}$ for all i. By Lemma 3.3, $(\frac{\partial f}{\partial x_i})^2$ has FTT-rank at most $\boldsymbol{r}^2$. To bound the FTT-rank of $\sum_{i=1}^d \left(\frac{\partial f}{\partial x_i}\right)^2$, the derivation is the same as in the proof of Lemma 3.1, only that the operator $\mathcal{D}$ is replaced by an operator mapping $\mathcal{C}^1(\mathbb{R}) \to \mathcal{C}(\mathbb{R})$ and $v \mapsto (\partial_x v)^2$. □

We now turn our view on the discretization of the corresponding operator with respect to $\Pi_{\boldsymbol{n}}$.

3.2.1 The operator in Tensor Train format

For a practical algorithm, we need a discretization of NonLin on the finite set $\Pi_{\mathbf{n}}$ such as (3.8) for the linear part. Here, we refrain from deriving a formula in the general setting of a full coefficient tensor and directly examine the case of a TT with fixed rank. We consider the square operation first. Let $\boldsymbol{n} \in \mathbb{N}^d$ and the *multiplication* operation $M_g\colon f \mapsto g \cdot f$, where g, f are given by

$$f(x) = v_{\boldsymbol{A}}(x) = \sum_{\boldsymbol{\alpha}} \boldsymbol{A}[\boldsymbol{\alpha}] \prod_{i=1}^d p^i_{\alpha_i}(x_i), \quad g(x) = v_{\boldsymbol{B}}(x) = \sum_{\boldsymbol{\beta}\in[\boldsymbol{n}]} \boldsymbol{B}[\boldsymbol{\beta}] \prod_{j=1}^d p^j_{\beta_j}(x_j) \tag{3.10}$$

with tensors $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{\boldsymbol{n}+1}$ both given in Tensor Train format with TT-rank $\boldsymbol{r} = (r_1,\ldots,r_{d-1}) \in \mathbb{N}^{d-1}$ and

$$\boldsymbol{A}[\boldsymbol{\alpha}] = A_1[\alpha_1]\cdots A_d[\alpha_d], \quad \boldsymbol{B}[\boldsymbol{\beta}] = B_1[\beta_1]\cdots B_d[\beta_d]. \tag{3.11}$$

We aim to define a Tensor Train operator $\boldsymbol{M}_{\boldsymbol{B}}\colon \mathbb{R}^{\boldsymbol{n}+1} \to \mathbb{R}^{2\boldsymbol{n}+1}$ such that

$$M_g(f) = v_{\boldsymbol{M}_{\boldsymbol{B}}(\boldsymbol{A})} = \sum_{\boldsymbol{\gamma}\in[2\boldsymbol{n}]} \boldsymbol{M}_{\boldsymbol{B}}(\boldsymbol{A})[\boldsymbol{\gamma}] \prod_{i=1}^d p^i_{\gamma_i}(x_i). \tag{3.12}$$

For $n \in \mathbb{N}_0$ let $T_{i,n} \in \mathbb{R}^{n+1,n+1}$ denote the transformation matrix mapping the coefficients of Legendre polynomials up to degree n on $[a_i, b_i]$ to the corresponding coefficients of standard monomials $1, x, x^2, \ldots$ up to degree n. Let

$$\hat{\boldsymbol{A}}[\boldsymbol{\alpha}] := \hat{A}_1[\alpha_1]\cdots\hat{A}_d[\alpha_d], \quad \hat{A}_i[\alpha_i] = \sum_{\alpha_i'=0}^{n_i} T_{i,n_i}[\alpha_i,\alpha_i']A_i[\alpha_i'], \tag{3.13}$$

$$\hat{\boldsymbol{B}}[\boldsymbol{j}] := \hat{B}_1[\beta_1]\cdots\hat{B}_d[\beta_d], \quad \hat{B}_i[\beta_i] := \sum_{\beta_i'=0}^{n_i} T_{i,n_i}[\beta_i,\beta_i']B_i[\beta_i'], \tag{3.14}$$

define the coefficient tensors of f and g with respect to monomials. Now, for $i = 1 \dots, d$ and $\alpha_i = 0, \dots, n_i$ define the matrix D_{i,α_i} by

$$D_{i,\alpha_i} := \left[\begin{array}{c} \mathbf{0}_{\alpha_i, n_i+1} \\ \hline I_{n_i+1, n_i+1} \\ \hline \mathbf{0}_{n_i+1-\alpha_i, n_i+1} \end{array}\right] \in \mathbb{R}^{2n_i+1, n_i+1}, \tag{3.15}$$

where $\mathbf{0}_{m,n} \in \mathbb{R}^{m,n}$ is a matrix with all entries equal to 0, which we define to be empty if m or n equal zero. Furthermore, for $i = 1, \dots, d$, $k_{i-1}, \ell_{i-1} \in \{1, \dots, r_{i-1}\}$, $k_i, \ell_i \in \{1, \dots, r_i\}$ we define the vector $\hat{\mathrm{C}}_i[k_{i-1}, \ell_{i-1}, k_i, \ell_i] \in \mathbb{R}^{2n_i+1}$ as

$$\hat{\mathrm{C}}_i[k_{i-1}, \ell_{i-1}, k_i, \ell_i] = \sum_{\alpha_i=0}^{n_i} D_{i,\alpha_i} \hat{B}_i[k_{i-1}, :, k_i] \hat{A}_i[\ell_{i-1}, \alpha_i, \ell_i]. \tag{3.16}$$

Note that $D_{i,\alpha_i} \hat{B}_i[k_{i-1}, :, k_i]$ denotes a matrix-vector multiplication, whereas $\hat{A}_i[\ell_{i-1}, \alpha_i, \ell_i]$ is scalar valued.

With slight abuse of notation, we denote the γ_i-th entry of the vector $\hat{\mathrm{C}}_i[k_{i-1}, \ell_{i-1}, k_i, \ell_i]$ by $\hat{\mathrm{C}}_i[k_{i-1}, \ell_{i-1}, \gamma_i, k_i, \ell_i] \in \mathbb{R}$, which defines an order 5 tensor $\hat{C}_i \in \mathbb{R}^{r_{i-1}, r_{i-1}, 2n_i+1, r_i, r_i}$. For convenience, we reshape $\hat{C}_i$ to an order 3 tensor by flattening together the first two and last two dimensions, again overloading notation with $\hat{C}_i \in \mathbb{R}^{r_{i-1}^2, 2n_i+1, r_i^2}$. Now we revert to the Legendre polynomial system and define the coefficient tensor $\boldsymbol{C} \in \mathbb{R}^{2\boldsymbol{n}+1}$ given in TT format by

$$\boldsymbol{C}[\boldsymbol{\gamma}] := C_1[\gamma_1] \cdots C_d[\gamma_d], \quad C_i[\gamma_i] = \sum_{\gamma_i'=0}^{2n_i} T_{i,2n_i}^{-1}[\gamma_i, \gamma_i'] \hat{C}_i[\gamma_i']. \tag{3.17}$$

This construction yields the following result.

Lemma 3.5. *Let f and g have FTT-rank $\boldsymbol{r}$ and given as in (3.10). Then, fg has FTT-rank at most $\boldsymbol{r}^2$, in particular*

$$g(x)f(x) = \sum_{\boldsymbol{\gamma} \in [2\boldsymbol{n}]} C[\boldsymbol{\gamma}] \prod_{i=1}^{d} p_{\gamma_i}^i(x_i), \tag{3.18}$$

with coefficient tensor $\boldsymbol{C}$ with TT-rank at most $\boldsymbol{r}^2$ given by (3.17).

By this Lemma, we have

$$\boldsymbol{M}_{\boldsymbol{B}}(\boldsymbol{A}) = \boldsymbol{C} \tag{3.19}$$

with $\boldsymbol{C}$ from (3.17). For ease of notation, we further define the *square* operation $\boldsymbol{S} \colon \mathbb{R}^{\boldsymbol{n}+1} \to \mathbb{R}^{2\boldsymbol{n}+1}$, $\boldsymbol{A} \mapsto \boldsymbol{M}_{\boldsymbol{A}}(\boldsymbol{A})$.

Finally, note that the partial derivative ∂_{x_i} defines a linear operator that, analogous to Section 3.1, implies a linear operator $\boldsymbol{L}_{x_i} \colon \mathbb{R}^{\boldsymbol{n}+1} \to \mathbb{R}^{\boldsymbol{n}+1}$ based on the polynomial discretization such that $\partial_{x_i} v_{\boldsymbol{A}} = v_{\boldsymbol{L}_{x_i} \boldsymbol{A}}$. This operator has the form $\boldsymbol{L}_{x_i} = \mathcal{I} \otimes \dots \otimes \mathcal{I} \otimes D_{x_i} \otimes \mathcal{I} \otimes \dots \otimes \mathcal{I}$ with

$D_{x_i} \in \mathbb{R}^{n_i+1,n_i+1}$ given in Appendix E.2. Putting all of the above together, we see that

$$\begin{aligned}\langle \nabla v_{\boldsymbol{B}}, \nabla v_{\boldsymbol{A}} \rangle &= \sum_{i=1}^{d} (\partial_{x_i} v_{\boldsymbol{A}})(\partial_{x_i} v_{\boldsymbol{B}}) = \sum_{i=1}^{d} v_{\boldsymbol{L}_{x_i}\boldsymbol{A}} v_{\boldsymbol{L}_{x_i}\boldsymbol{B}} \\ &= \sum_{i=1}^{d} v_{\boldsymbol{M}_{\boldsymbol{L}_{x_i}\boldsymbol{B}}(\boldsymbol{L}_{x_i}\boldsymbol{A})} = v_{\sum_{i=1}^{d} \boldsymbol{M}_{\boldsymbol{L}_{x_i}\boldsymbol{B}}(\boldsymbol{L}_{x_i}\boldsymbol{A})}\end{aligned} \tag{3.20}$$

This leads to a Tensor Train operator representing the nonlinear part (2.8). In particular, for $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{\boldsymbol{n}+1}$ let

$$\boldsymbol{NL}(\boldsymbol{A}) \coloneqq -\sum_{i=1}^{d} \boldsymbol{S}(\boldsymbol{L}_{x_i}\boldsymbol{A}), \tag{3.21}$$

$$\boldsymbol{NL}_{\boldsymbol{B}}(\boldsymbol{A}) \coloneqq -\sum_{i=1}^{d} \boldsymbol{M}_{\boldsymbol{L}_{x_i}\boldsymbol{B}}(\boldsymbol{L}_{x_i}\boldsymbol{A}). \tag{3.22}$$

Then, by (3.20), we have $\mathrm{NonLin}(v_{\boldsymbol{A}}) = v_{\boldsymbol{NL}(\boldsymbol{A})}$ and $-\langle \nabla v_{\boldsymbol{B}}, \nabla v_{\boldsymbol{A}} \rangle = v_{\boldsymbol{NL}_{\boldsymbol{B}}(\boldsymbol{A})}$. This concludes the derivation of the nonlinear part.

3.3 Projection and retraction

The discussion so far shows that linear and nonlinear operations on the polynomial discretization with Tensor Trains may increase the rank as well as the underlying polynomial degree. Therefore, we shall discuss operations that keep a fixed polynomial degree and a fixed TT-rank with possible error control, namely *projection* and *retraction*. Regarding the projection, let $\boldsymbol{n}, \boldsymbol{m} \in \mathbb{N}_0$, $\boldsymbol{n} \leq \boldsymbol{m}$ and define $\mathcal{P}_{\boldsymbol{m},\boldsymbol{n}} \colon \operatorname{span} \Pi_{\boldsymbol{m}} \to \operatorname{span} \Pi_{\boldsymbol{n}}$ by

$$\mathcal{P}_{\boldsymbol{m},\boldsymbol{n}}(\cdot) \coloneqq \sum_{\alpha_1,\dots,\alpha_d=0}^{n_1,\dots,n_d} \bigotimes_{i=1}^{d} p_{\alpha_i}^i \left\langle \bigotimes_{i=1}^{d} p_{\alpha_i}^i, \cdot \right\rangle \tag{3.23}$$

Due to the orthonormality of the $p_{\alpha_i}^i$, the projection is simply obtained by truncating the coefficients, as the following result states.

Lemma 3.6. *For $\boldsymbol{n} \leq \boldsymbol{m}$ and $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{m}+1}$ we have $\mathcal{P}_{\boldsymbol{m},\boldsymbol{n}} V_{\boldsymbol{A}} = V_{\boldsymbol{P}_{\boldsymbol{m},\boldsymbol{n}}\boldsymbol{A}}$, where $\boldsymbol{P}_{\boldsymbol{m},\boldsymbol{n}} \colon \mathbb{R}^{\boldsymbol{m}+1} \to \mathbb{R}^{\boldsymbol{n}+1}$ is defined by*

$$(\boldsymbol{P}_{\boldsymbol{m},\boldsymbol{n}}\boldsymbol{A})[\alpha_1,\dots,\alpha_d] = \boldsymbol{A}[\alpha_1,\dots,\alpha_d] \tag{3.24}$$

for all $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{m}+1}$ and $\boldsymbol{\alpha} \in \mathbb{N}_0^{\boldsymbol{n}}$.

Note that by Parseval's identity, the projection error in L^2-norm can be computed by simply adding the squares of the elements that are eliminated by the projection, i.e. with assumptions of Lemma 3.6, we have

$$\|\mathcal{P}_{\boldsymbol{m},\boldsymbol{n}} v_{\boldsymbol{A}} - v_{\boldsymbol{A}}\|_{L^2(K)}^2 = \sum_{\alpha_1=n_1+1,\dots,\alpha_d=n_d+1}^{m_1,\dots,m_d} \boldsymbol{A}[\alpha_1,\dots,\alpha_d]^2. \tag{3.25}$$

A possible realization of a *retraction operator*

$$\boldsymbol{R}_{\boldsymbol{r}}\colon \bigcup_{\hat{\boldsymbol{r}}\geq \boldsymbol{r}} \mathcal{M}_{\hat{\boldsymbol{r}}} \to \mathcal{M}_{\boldsymbol{r}}, \tag{3.26}$$

for given fixed rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$, is obtained by using the *TT rounding* scheme first presented in [29, Algorithm 2], which is based on efficient high-order singular value decomposition exploiting the structure of TTs. The operators in (3.24) and (3.26) provide us with the necessary tensor operations to fix the degree as well as the rank of the HJB solution, concluding this section.

4 A direct low-rank HJB solver

In this section, we consider polynomial potentials $\Phi \in \operatorname{span} \Pi_{\boldsymbol{n}}$ for some $\boldsymbol{n} \in \mathbb{N}_0^d$. If the potential is not available in polynomial form, we can obtain a suitable polynomial approximation e.g. by the *Alternating Linear Scheme* (ALS) [19] as was done in [31] for the purpose of approximating value functions. Crucially, the ALS yields an approximation in a chosen low rank TT format. For $\Phi \in \operatorname{span} \Pi_{\boldsymbol{n}}$, we consider a projected version of the modified HJB equation (2.9) restricted to the hypercube K defined by

$$\begin{cases} \partial_t v_t &= \mathcal{P}_{2\boldsymbol{n},\boldsymbol{n}}\left[\mathrm{Lin}(v_t) + \mathrm{NonLin}(v_t)\right], \\ v_0 &= \Phi, \end{cases} \qquad \text{in } K, \tag{4.1}$$

for $t \in [0,T]$ and some $T > 0$ large enough. Note, that the projection only acts on the nonlinear part, as the linear part does not increase the polynomial degree.

With the work from the previous section, we can show that this PDE is equivalent to an ODE on a tensor space. Let $\boldsymbol{L}$, $\boldsymbol{NL}$, $\boldsymbol{NL}_{\boldsymbol{B}}$ for any $\boldsymbol{B} \in \mathbb{R}^{\boldsymbol{n}+1}$ and $\boldsymbol{P} \equiv \boldsymbol{P}_{2\boldsymbol{n}+1,\boldsymbol{n}+1}$ be given by (3.8), (3.21), (3.22) and (3.24), respectively. Then the following theorem holds true.

Theorem 4.1 (Projected HJB equation is equivalent to tensor-valued ODE)**.** *Let $\boldsymbol{A}(t) \in \mathbb{R}^{\mathbf{n}+1}$ be a solution of the tensor-valued ODE*

$$\dot{\boldsymbol{A}}(t) = \boldsymbol{L}\boldsymbol{A}(t) + \boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}(\boldsymbol{A}(t)), \qquad \boldsymbol{A}(0) = \boldsymbol{A}_0, \tag{4.2}$$

for $t \in [0,T]$. Then $v_t := v_{\boldsymbol{A}(t)}$ solves (4.1). Conversely, if $v_t \in \operatorname{span} \Pi_{\boldsymbol{n}}$ solves (4.1), then there exists a unique $\boldsymbol{A}(t) \in \mathbb{R}^{\mathbf{n}+1}$ such that $v_t = v_{\boldsymbol{A}(t)}$ and $\boldsymbol{A}(t)$ solves (4.2).

Proof. Let $\boldsymbol{A}(t) \in \mathbb{R}^{\mathbf{n}+1}$ solve (4.2). Then, $\dot{v}_{\boldsymbol{A}(t)} = v_{\dot{\boldsymbol{A}}(t)} = v_{\boldsymbol{L}\boldsymbol{A}(t)+\boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}(A(t))} = v_{\boldsymbol{L}\boldsymbol{A}(t)} + v_{\boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}(\boldsymbol{A}(t))} = \mathrm{Lin}(v_{\boldsymbol{A}(t)}) + \mathcal{P}_{2\boldsymbol{n}+1,\boldsymbol{n}+1}\left[\mathrm{NonLin}(v_{\boldsymbol{A}(t)})\right]$ and $v_{\boldsymbol{A}(0)} = v_{\boldsymbol{A}_0} = \Phi$, showing the first part of the claim. Conversely, if $v_t \in \operatorname{span} \Pi_{\boldsymbol{n}}$ solves (4.1), then there exists a unique $\boldsymbol{A}(t)$ with $v_t = v_{\boldsymbol{A}(t)}$ and $v_{\dot{\boldsymbol{A}}(t)} = \partial_t v_t = v_{\boldsymbol{L}\boldsymbol{A}(t)+\boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}(\boldsymbol{A}(t))}$. Since the mapping $\boldsymbol{A} \mapsto v_{\boldsymbol{A}}$ is injective, this yields the second part of the claim. □

The solution algorithm for (4.2) which will be presented in the following relies on local linearizations of the HJB for stiffnes based stepsize control. Hence, we state the following result on the form of such local linearizations.

Lemma 4.1 (Local linearization)**.** *Let $\boldsymbol{B} \in \mathbb{R}^{\mathbf{n}+1}$. Then, the linearization of (4.2) at $\boldsymbol{B}$ is given by*

$$\dot{\boldsymbol{A}}(t) = (\boldsymbol{L} + 2\boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}_{\boldsymbol{B}})\boldsymbol{A}(t) - \boldsymbol{P}\boldsymbol{N}\boldsymbol{\mathcal{L}}(\boldsymbol{B}). \tag{4.3}$$

Proof. Note that the linearization of $\mathrm{NonLin}(v) = -\|\nabla v\|^2$ around a fixed $v_0 \in \operatorname{span} \Pi_{\mathbf{n}}$ is given by

$$\mathrm{NonLin}_{v_0}(v) = -2\langle \nabla v_0, \nabla v\rangle + \|\nabla v_0\|^2 = -2\langle \nabla v_0, \nabla v\rangle - \mathrm{NonLin}(v_0). \tag{4.4}$$

Now, for $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{\mathbf{n}+1}$ we have

$$\begin{aligned}\mathrm{NonLin}_{v_{\boldsymbol{B}}}(v_{\boldsymbol{A}}) &= -2\langle \nabla v_{\boldsymbol{B}}, \nabla v_{\boldsymbol{A}}\rangle - \mathrm{NonLin}(v_{\boldsymbol{B}}) &(4.5)\\ &= 2v_{\boldsymbol{NL}_{\boldsymbol{B}}\boldsymbol{A}} - v_{\boldsymbol{NL}(\boldsymbol{B})} &(4.6)\\ &= v_{2\boldsymbol{NL}_{\boldsymbol{B}}\boldsymbol{A}-\boldsymbol{NL}(\boldsymbol{B})}. &(4.7)\end{aligned}$$

Since the other operators appearing on the right hand side of (4.1) are linear, (4.3) follows. □

By Theorem 4.1, it suffices to solve (4.2) for $\boldsymbol{A}(t)$ since this solution defines the solution of (4.1) via $v_t = v_{\boldsymbol{A}(t)}$. In the rest of this section, we present principled ways of computing approximate solutions to (4.2) on the low rank manifold $\mathcal{M}_{\mathbf{r}}$. Two methods are investigated:

1 A simple explicit Euler scheme with adaptive step sizes and retraction after every step, see Section 4.1.

2 A dynamical low rank integrator designed for time integration of Tensor Trains [26], see Section 4.2.

4.1 Time adaptive explicit Euler scheme

Preliminaries. In the following, we define a number of time points $N \in \mathbb{N}$, a sequence of times $0 = t_0 < t_1 < \ldots < t_N = T$, a TT-rank function $t \mapsto \boldsymbol{r}_t \in \mathbb{N}^{d-1}$ assigning to every time a Tensor Train rank and discrete approximations $\mathcal{M}_{\mathbf{r}_{t_n}} \ni \boldsymbol{Y}_{t_n} \approx \boldsymbol{A}(t_n), n = 0, \ldots, N$, to the solution $\boldsymbol{A}(t)$ of (4.2). Throughout this section, let $\tau_{\max}, \delta_{\mathrm{proj}}, \delta_{\mathrm{rank}}, \delta_{\mathrm{contr}} > 0$ and a reduction parameter $\rho \in (0, 1)$. Denote the potential of the standard normal distribution by $v_\infty(x) = \|x\|^2/2$ and note that by Lemma D.2 this function has FTT rank $(2, \ldots, 2)$. In practice we choose $\boldsymbol{r}_t$ to be bounded by $\mathrm{TT\text{-}rank}(v_0)$ and $\mathrm{TT\text{-}rank}(v_\infty)$ with adaptive rank reduction based on TT-rounding error induced by the retraction from (3.26).

Time adaptive explicit Euler step. Starting with $n = 0$, we have $\boldsymbol{Y}_{t_n} \in \mathcal{M}_{\boldsymbol{r}_{t_n}}$. By Section 3, the right-hand side of (4.2) applied to $\boldsymbol{Y}_{t_n}$, i.e. the tensor $\boldsymbol{L}\boldsymbol{Y}_{t_n} + \boldsymbol{P}\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})$ has TT-rank at most $2\boldsymbol{r} + 2\boldsymbol{r}^2$ and so the addition

$$\overline{\boldsymbol{Y}}_{t_n+\tau_n} = \boldsymbol{Y}_{t_n} + \tau_n(\boldsymbol{L}\boldsymbol{Y}_{t_n} + \boldsymbol{P}\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})) \tag{4.8}$$

has TT-rank at most $3\mathbf{r} + 2\mathbf{r}^2$ for any $\tau_n > 0$.

Since we require the next iterate to be a Tensor Train of rank $\mathbf{r}_{t_n+\tau_n}$, we retract to the appropriate manifold, setting

$$\boldsymbol{Y}_{t_n+\tau_n} = \boldsymbol{R}_{\boldsymbol{r}_{t_n+\tau_n}}(\overline{\boldsymbol{Y}}_{t_n+\tau_n}), \tag{4.9}$$

where $\boldsymbol{R}_{\boldsymbol{r}}$ denotes the TT rounding procedure based on higher order singular value decomposition and mapping to $\mathcal{M}_{\boldsymbol{r}}$, which was presented in [29, Algorithm 2]. Note that (4.9) corresponds to (2.11) with the $\mathrm{Compression}$ given by the retraction operator, i.e. by the higher order singular value decomposition. Up to now the choice of the step size τ_n was arbitrary. In what follows we set constraints on the step size τ_n based on three stability criteria.

Criterion 1: local stiffness. At each iteration, we restrict the stepsize dependent on the local stiffness of the ODE. We use a heuristic based on local linearizations of (4.2) to determine suitable upper bounds for the stepsize. By Lemma 4.1, the local stiffness at the current iterate $\boldsymbol{Y}_{t_n}$ is governed by the linear operator

$$\boldsymbol{H}_{\boldsymbol{Y}_{t_n}} \coloneqq \boldsymbol{L} + 2\boldsymbol{P}\boldsymbol{N}\boldsymbol{L}_{\boldsymbol{Y}_{t_n}}. \tag{4.10}$$

If the current iterate $\boldsymbol{Y}_{t_n}$ defines a zero mean Gaussian with diagonal covariance $\operatorname{diag}(a_{ii}, i = 1, \dots, d)$, the eigenvalues of $\boldsymbol{H}_{\boldsymbol{Y}_{t_n}}$ can be bounded by $2\sum_{i=1}^{d} |1 - 2a_{ii}|$ (the details of the calculation can be found in Appendix E.3). In general, $\boldsymbol{H}_{\boldsymbol{Y}_{t_n}}$ defines a non-symmetric TT operator. To the knowledge of the authors, estimation of the largest absolute eigenvalue of general non-symmetric TT operators is an open question. Here, we rely on a simpler idea. In particular as we are dealing with real valued tensors $\boldsymbol{Y}_{t_n}$, we avoid analyzing the operator action on complex space. In contrast, we are interested in the effect of the current operator in the neighborhood of the current iterate. This is realized by estimating the largest absolute *real* eigenvalue of $\boldsymbol{H}_{\boldsymbol{Y}_{t_n}}$ denoted by λ_{t_n} with corresponding eigenspace that is not orthogonal to the current iterate $\boldsymbol{Y}_{t_n}$, by a power iteration. The resulting scheme is detailed in Algorithm 1. The current Tensor Train iterate $\boldsymbol{Y}_{t_n}$ serves as an initial guess for the eigentensor. The procedure then resembles a standard power iteration with an additional retraction step in line 6, which reduces computational burden. In practice we are only interested in the absolute value of the eigenvalue or a meaningful upper bound $\overline{\lambda}_{t_n}$ and not in the corresponding eigentensor. Note that the eigenvalue usually converges at much higher order than the eigentensor. The aforementioned upper bound then is obtained through a simple rounding up strategy with a specified number of accurate non-zero digits, see Algorithm 1. Based on the return $\overline{\lambda}_{t_n}$ of the power iteration, we define a maximal stable stepsize τ_λ by

$$\tau_\lambda \coloneqq \frac{2\rho}{|\overline{\lambda}_{t_n}|}. \tag{4.11}$$

In experiments, this stiffness estimation proves essential for the solver to converge.

Algorithm 1 Upper bound estimating the principal real eigenvalue λ_{t_n} of $\boldsymbol{H}_{\boldsymbol{Y}_{t_n}}$ from (4.10) based on power iteration.

Input:
- current iterate $\boldsymbol{X}_0 = \boldsymbol{Y}_{t_n}$
- application of $\boldsymbol{H}_{Y_{t_n}}$
- maximum allowed TT rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$
- number of correct non-zero digits $p \in \mathbb{N}$

Output: upper bound $\overline{\lambda}_{t_n}$

1: Let $\lambda_{t_n}^k$ denote the k-th iterate.
2: Set $k = 0$.
3: **while** p-th non-zero digit of $\lambda_{t_n}^k$ is changing **do**
4: Let $K \in \mathbb{N}$, $\boldsymbol{X}_0 \in \mathcal{M}_{\mathbf{r}}$.
5: $\hat{\boldsymbol{X}}_k = \boldsymbol{X}_k / \|\boldsymbol{X}_k\|_F$
6: $\boldsymbol{X}_{k+1} = \boldsymbol{R}_{\mathbf{r}}(\boldsymbol{H}_{\boldsymbol{Y}_{t_n}} \hat{\boldsymbol{X}}_k)$
7: $\lambda_{t_n}^k = \langle \hat{\boldsymbol{X}}_k, \boldsymbol{X}_{k+1} \rangle$
8: $k = k + 1$
9: **end while**
10: Define position P of first non zero digit with $P = \lceil -\log_{10}(\lambda_{t_n}^k) \rceil$.
11: Define upper bound treshold $\epsilon_p = 10^{-(P+p)}$.
12: Define $\overline{\lambda}_{t_n}^k = \lambda_{t_n}^k + \epsilon_p$.

Criterion 2: local relative projection error. For stepsize $\tau > 0$ consider the iterate $\overline{\boldsymbol{Y}}_{t_n+\tau}$ defined by (4.8) for $\tau_n = \tau$ and let $\overline{\overline{\boldsymbol{Y}}}_{t_n+\tau} = \boldsymbol{Y}_{t_n} + \tau(\boldsymbol{L}\boldsymbol{Y}_{t_n} + \boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n}))$ be an Euler step with the non-projected equation. Let

$$\tau_{\mathrm{proj}} \coloneqq \begin{cases} \tau_{\max}, & \text{if } \|\boldsymbol{P}\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n}) - \boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})\|_F = 0, \\ \frac{\delta_{\mathrm{proj}}}{\|\boldsymbol{P}\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n}) - \boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})\|_F / \|\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})\|_F}, & \text{else.} \end{cases} \tag{4.12}$$

Then, for any $\tau \leq \tau_{\mathrm{proj}}$ we get

$$\|\overline{\boldsymbol{Y}}_{t_n+\tau} - \overline{\overline{\boldsymbol{Y}}}_{t_n+\tau}\|_F \leq \tau \|\boldsymbol{P}\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n}) - \boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})\|_F \leq \delta_{\mathrm{proj}} \|\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})\|_F. \tag{4.13}$$

Hence, the projection error of the Euler step, normalized with respect to the magnitude of the degree increasing nonlinear part $\boldsymbol{N}\boldsymbol{L}(\boldsymbol{Y}_{t_n})$, is bounded from above by δ_{proj}.

Criterion 3: local relative retraction error. Determine maximum τ_{rank} such that

$$\frac{\|\overline{\boldsymbol{Y}}_{t_n+\tau_{\mathrm{rank}}} - \boldsymbol{Y}_{t_n+\tau_{\mathrm{rank}}}\|_F}{\|\overline{\boldsymbol{Y}}_{t_n+\tau_{\mathrm{rank}}}\|_F} \leq \delta_{\mathrm{rank}}. \tag{4.14}$$

Here, we initially choose $\tau^0_{\mathrm{rank}} = \tau_{n-1}$ and proceed with $\tau^k_{\mathrm{rank}} = \frac{1}{2}\tau^{k-1}_{\mathrm{rank}}$ until τ^k_{rank} fulfils condition (4.14). Then, we use bisection iteration to determine the maximum $\tau_{\mathrm{rank}} \in (\frac{1}{2}\tau^k_{\mathrm{rank}}, \tau^k_{\mathrm{rank}}]$ satisfying (4.14).

Final stepsize choice. After these three criteria, the next step size τ_n in (4.9) and the next time t_{n+1} are defined as

$$\tau_n \coloneqq \min\{\tau_{\max}, \tau_\lambda, \tau_{\mathrm{proj}}, \tau_{\mathrm{rank}}, T - t_n\}, \tag{4.15}$$

$$t_{n+1} \coloneqq t_n + \tau_n, \tag{4.16}$$

where the term $T - t_n$ ensures that we end exactly at terminal time T. The single time step (4.9) is repeated for $n = 0, 1, \ldots$ with stepsize (4.15) until $t_{n+1} = T$, in which case we define $N = n + 1$.

In addition to the adaptivity in the stepsize, the solver also incorporates adaptivity in the polynomial degree as well as the TT rank, which is detailed in the following.

Adaptive decrease of polynomial degree Motivated by the fact that $v_t \to v_\infty \in \Pi_{(2,\ldots,2)}$ as $t \to \infty$ at exponential rate, we introduce a simple adaptive choice for the polynomial degree. Assume that the degrees of $\boldsymbol{Y}_{t_n}$ at time t_n are given by $\boldsymbol{n}_{t_n} \in \mathbb{N}_0^d$. Let $\boldsymbol{Y}^k_{t_n}$ denote the order $d-1$ tensor, which for $k = 1, \ldots, d$ is given as

$$\boldsymbol{Y}^k_{t_n} = (\boldsymbol{Y}_{t_n}[\boldsymbol{\alpha}])_{\boldsymbol{\alpha} \in [\boldsymbol{n}_{t_n}], \alpha_k = (\boldsymbol{n}_{t_n})_k}.$$

This is a slice of the full coefficient tensor $\boldsymbol{Y}_{t_n}$ fixing $\alpha_k = (\boldsymbol{n}_{t_n})_k$ which is the highest polynomial degree in the k-th dimension at time t_n. Now, in case of

$$\|\boldsymbol{Y}^k_{t_n}\|_F \leq \delta_{\mathsf{contr}}, \tag{4.17}$$

we truncate the highest polynomial degree in the k-th direction. Since $\boldsymbol{Y}_{t_n}$ is given in TT format with

$$\boldsymbol{Y}_{t_n}[\boldsymbol{\alpha}] = Y_{t,1}[\alpha_1] \cdot \ldots \cdot Y_{t,d}[\alpha_d],$$

this operation is realized by truncation of the component tensor Y_k and possibly adapting the TT-ranks.

Adaptive choice of TT rank Motivated by the conjecture, that the FTT rank of v_t is bounded by the FTT rank of v_0 and v_∞, i.e. $\boldsymbol{r}_\infty = (2, \dots, 2)$, we perform two retraction steps with respect to these bounds after the time step at time t_n. First a retraction with respect to the rank

$$\hat{\boldsymbol{r}}_{t_n+\tau_n} = \max\{\boldsymbol{r}_{t_n}, \boldsymbol{r}_\infty\} \tag{4.18}$$

is performed where the maximum is understood component wise. This serves to ensure that the rank of the solution remains bounded by the maximum of the initial rank and the rank of the standard normal potential. Furthermore, a rounding procedure [29, Algorithm 2] with respect to δ_{contr} is performed to potentially further decrease the rank and thus define $\boldsymbol{r}_{t_n+\tau_n}$. In practice both retraction steps can be performed efficiently in a single operation, which leads to (4.9).

The proposed time adaptive explicit Euler scheme is summarized in Algorithm 2.

Algorithm 2 Time adaptive explicit Euler Scheme to solve HJB equation based on Tensor Trains

Input:
- v_0 given in TT format,
- $T > 0$ maximum finite time horizon,
- $\tau_{\text{max}} > 0$, bound for the stepsize
- reduction stiffness parameter $\rho \in (0, 1)$,
- step size proposal hyperparameter $\delta_{\text{proj}}, \delta_{\text{rank}}$,
- degree of freedom contribution tolerance $\delta_{\text{contr}} > 0$.

Output: Discrete sequence $(v_{t_n})_n$ defined on subsequently determined adaptive time points $t_n \in [0, T]$.

Set $t = 0$.
while $t \leq T$ **do**
  Determine next time step:
    Compute maximal stable stepsize τ_λ. ▷ see (4.11)
    Compute step size proposal τ_{proj} based on projection error. ▷ see (4.12)
    Compute step size proposal τ_{rank} based on relative retraction error. ▷ see (4.14)
    Determine final step size $\tau = \min\{\tau_{\max}, \tau_\lambda, \tau_{\text{proj}}, \tau_{\text{rank}}, T - t\}$.
  Perform a single Euler step
    Set $t = t + \tau$.
    Approximate v_t via algebraic manipulation of the underlying TT format. ▷ see (4.8)
    Perform a retraction step of the resulting coefficient in TT format. ▷ see (4.9)
  (Re-)compression
    Check for potential polynomial degree decrease using δ_{contr}. ▷ see (4.17)
    Check for potential rank reduction using δ_{contr}. ▷ see (4.18)
end while

4.2 Dynamical low rank approximation

While the time adaptive explicit Euler scheme presented in the previous section offers a conceptually simple integration method, *Dynamical low rank appromxation* (DLRA) [23, 25, 26] methods offer another principled way of approximately integrating tensor valued ODEs of the form (4.2).

Here, the idea is to formulate an approximation of a tensor valued ODE

$$\dot{\boldsymbol{A}}(t) = \boldsymbol{F}(t, \boldsymbol{A}(t)), \qquad \boldsymbol{A}(0) = \boldsymbol{A}_0,$$

where $\boldsymbol{n} \in \mathbb{N}^d$, $\boldsymbol{A}(t) \in \mathbb{R}^{\boldsymbol{n}}$ and $\boldsymbol{F}\colon [0,\infty] \times \mathbb{R}^{\boldsymbol{n}} \to \mathbb{R}^{\boldsymbol{n}}$ on a fixed rank manifold $\mathcal{M}_{\boldsymbol{r}}$. This is done via projection of the right-hand side onto the tangent space of $\mathcal{M}_{\boldsymbol{r}}$. More precisely, for a fixed $\boldsymbol{r} \in \mathbb{N}^{d-1}$, the approximation is defined as

$$\dot{\boldsymbol{Y}}(t) = \mathcal{P}_{\mathcal{T}_{\boldsymbol{Y}(t)}} \boldsymbol{F}(t, \boldsymbol{Y}(t)), \qquad \boldsymbol{Y}(0) = \boldsymbol{Y}_0 \approx \boldsymbol{A}_0, \tag{4.19}$$

where $\boldsymbol{Y}_0 \in \mathcal{M}_{\boldsymbol{r}}$ and $\mathcal{P}_{\mathcal{T}_{\boldsymbol{Y}(t)}}$ denotes the orthogonal projection (in Frobenius norm) onto the tangent space of $\mathcal{M}_{\boldsymbol{r}}$ in $\boldsymbol{Y}(t)$. Note that due to this projection, a solution of (4.19) satisfies $\boldsymbol{Y}(t) \in \mathcal{M}_{\boldsymbol{r}}$ for all t. In [13] the authors use an explicit Euler discretization of (4.19) for the solution of HJB equations appearing in deterministic optimal control based on spatially discretized parabolic PDEs. However, leveraging the form of the tangent space, the projector on the right hand side can be decomposed into a sum of projectors corresponding to orthogonal subspaces. In [25], the authors propose to use this sum structure for a Lie-Trotter type splitting scheme in the case of a matrix valued ODE, which is termed the *projector-splitting* integrator. Consequently, [26] extends the projector splitting integrator to the tensor setting. One of the key properties of this integrator is that each discrete step preserves the rank $\boldsymbol{r}$.

In our scheme, using a step with the integrator from [26] instead of the explicit Euler step (4.8) leads to a new iterate $\overline{\boldsymbol{Y}}_{t_n+\tau_n}$ with the same rank as $\boldsymbol{Y}_{t_n}$. Hence, the retraction (4.9) becomes a mere rounding procedure and the rank of two consecutive iterates is monotonically decreasing. This is a desirable property if the initial rank satisfies $\boldsymbol{r}_{t_0} \geq (2,\dots,2)$. For $\boldsymbol{r}_{t_0} \ngeq (2,\dots,2)$, the projector-splitting is unsuited because it restricts the rank from above to $\boldsymbol{r}_{t_n} \leq \boldsymbol{r}_{t_0}$ and so $\boldsymbol{r}_{t_n}$ can not converge to the correct rank $(2,\dots,2)$.

Incorporating more recent state-of-the-art dynamical low rank integrators for matrix valued ODEs such as [8, 9] to the Tensor Train setting could lead to significant improvements of the proposed method. In particular, the Basis Update $\&$ Galerkin (BUG) integrator [8] introduces rank adaptivity, while the fully parallel integrator [9] could additionally greatly speed up computations in high dimensions. However, to the knowledge of the authors neither of these integrators have been formulated for the setting of high dimensional Tensor Trains at the time of writing. Therefore, their application in our method remains a topic of future research.

4.3 Evaluation of the low-rank model

As the result of section 4.1 or 4.2 we have a representation of the value function in the spirit of (3.5) at discrete set of time points the form $t \in \{t_0, t_1, \dots, T\}$ of the form

$$v_t(x) = \sum_{\boldsymbol{\alpha} \in [\boldsymbol{n}_t]} Y_t[\boldsymbol{\alpha}] p_{\boldsymbol{\alpha}}(x), \tag{4.20}$$

for some $\boldsymbol{n}_t \in \mathbb{N}_0^d$ and Y_t given in tensor train format resulting as the discrete solution of (4.2).

We now want to discuss the evaluation of $v_t(x)$ at arbitrary time $t \in [0,T]$ and $x \in \mathbb{R}^d$. This is motivated by the reverse-time sampling process, which is permitted to be time adaptive and may require evaluation in time points not included in the set $\{t_0,\dots,t_N\}$.

For this we propose a very simple solution. Let $t^* \in [0,T]$. Let

$$\bar{t} = \max\{t \in \{t_0,\dots,t_N\}\colon t \leq t^*\} \tag{4.21}$$

Let $\tau = t^* - \underline{t}$. Then, we compute the coefficient representation in Tensor train format of v_{t^*} through a single Euler- or DLRA step with step size τ. Note that this step size is within the step size bounds

implied by the adaptive scheme proposed earlier. In particular, τ is smaller than the step size implied by local stiffness.

Lastly, we discuss how the evaluation of the model class is performed in practice. Aside from the evaluation of the polynomial basis functions, only matrix- and vector products have to be computed to evaluate v_t. This efficient evaluation is one of the strengths of the Tensor Train format. For $x = (x_s, \dots, x_d) \in \mathbb{R}^d$ and $t \in [0, T]$, the approximation is defined by a TT $\mathbf{Y}_t$ with dimensions $\boldsymbol{n}_t = (n_{t,1}, \dots, n_{t,d})$ and ranks $\mathbf{r}_t = (r_{t,1}, \dots, r_{t,d})$. To evaluate (4.20), one first computes $p_j^i(x_i)$ for all $i = 1, \dots, d$ and $j = 0, \dots, n_{t,i}$. Now the TT format provides a decomposition of the form $\mathbf{Y}_t[\boldsymbol{\alpha}] = Y_{t,1}[\alpha_1] Y_{t,2}[\alpha_2] \cdots Y_{t,d}[\alpha_d]$, where $Y_{t,i} \in \mathbb{R}^{r_{t,i-1}, n_{t,i}, r_{t,i}}$, $\boldsymbol{\alpha} \in [\boldsymbol{n}_t]$ and hence α_i runs from 0 to $n_{t,i}$. In particular, (4.20) implies

$$\begin{aligned} v_t(x) &= \sum_{\alpha_1=0}^{n_{t,1}} \dots \sum_{\alpha_d=0}^{n_{t,d}} Y_{t,1}[\alpha_1] Y_{t,2}[\alpha_2] \cdots Y_{t,d}[\alpha_d] p_{\alpha_1}^1(x_1) p_{\alpha_2}^2(x_2) \dots p_{\alpha_d}^d(x_d) \\ &=: Y_{t,1}^{x_1} \cdot Y_{t,2}^{x_2} \cdots Y_{t,d}^{x_d}, \end{aligned} \tag{4.22}$$

where $Y_{t,i}^{x_i} \in \mathbb{R}^{r_{t,i-1}, r_{t,i}}$ results from a simple contraction of $Y_{t,i}$ with the vector $(p_1^i(x_i), \dots, p_{n_{t,i}}^i(x_i))$ over the $n_{t,i}$-dimension. $Y_{t,1}^{x_1} \cdot Y_{t,2}^{x_2} \cdots Y_{t,d}^{x_d}$ is now a simple matrix product. Note since $r_{t,0} = r_{t,d} = 1$, this product boils down to a matrix-vector product, when performed from left to right or vice-versa, yielding a scalar value.

5 Numerical results

Based on Remark 2.1, we generate approximate samples from μ_* by means of the discrete process described in Algorithm 3. The algorithm utilizes the reverse-time process from Remark 2.1 with $\lambda = 0$ discretized at the time-points t_n at which approximate solutions $\boldsymbol{Y}_{t_n}$ of the projected HJB (4.1) are available. These approximations define our surrogate for the score $\nabla \log \pi_t$ based on

$$v_{t_n} \approx -\log \pi_{t_n}, \quad n = 0, \dots, N, \tag{5.1}$$

where $v_{t_n} \coloneqq v_{\boldsymbol{Y}_{t_n}}$ is understood in the sense of (3.5). The inner loop over k in Algorithm 3 consists of additional *Langevin-postprocessing* steps [39] after every step with the reverse process.

As a necessary condition for convergence of the computed solutions v_{t_n} to the potential $v_\infty(x) = \frac{1}{2} x^\intercal I_d x$ of the standard normal distribution, we consider convergence of the coefficients of the quadratic part. More precisely, since v_{t_n} is a polynomial, we can always write

$$v_{t_n}(x) = a_{t_n} + b_{t_n}^\intercal x + x^\intercal \Sigma_{t_n}^{-1} x + \text{higher order terms}, \tag{5.2}$$

with $a_{t_n} \in \mathbb{R}$, $b_{t_n} \in \mathbb{R}^d$ and a symmetric $\Sigma_{t_n} \in \mathbb{R}^{d \times d}$. In this section, we call *covariance error at time* t_n the term

$$\mathrm{CovErr}(t_n) = \left\| \Sigma_{t_n}^{-1} - I_d/2 \right\|_F / \left\| I_d/2 \right\|_F, \tag{5.3}$$

i.e. the relative error in Frobenius norm between the current precision matrix and the precision matrix of the standard normal distribution.

We remark that, in the test cases we considered, the results produced by the dynamical low rank integrator [26] (using the same heuristics for adaptive stepsize determination) are similar to the results produced by an explicit Euler stepping with subsequent retraction. Hence, we only present the results of the latter.

Algorithm 3 Sampling from π_*

Input:
- Initial samples $\{z_0^i\}_{i=1}^I \sim \mathcal{N}(0, I_d)$,
- Times $\{t_n\}_{n=1}^N$ and discrete HJB solution $\{v_{\boldsymbol{Y}_{t_n}}\}_{n=1}^N$ defined by Algorithm 2,
- Stepsize τ and number of steps $K \in \mathbb{N}$ for Langevin postprocessing,
- Parameter $\lambda \in [0, 1]$ for reverse-time process

Output: Samples $\{z_N^i\}_{i=1}^I \sim \mu_*$.

for $i = 1, \dots, I$ in parallel **do**
  Generate time points $\{t_n^i\}_{n=1}^N$.
  for $n = 0, 1, \dots, N-1$ **do**
    Set $\tau_n^i = t_{n+1}^i - t_n^i$
    Sample $\xi_n \sim \mathcal{N}(0, I_d)$ if $\lambda \neq 1$.
    Set $z_{n+1}^i = z_n^i + \left[z_n^i + (2-\lambda)\nabla v_{\boldsymbol{Y}_{T-t_n}}(z_n^i)\right]\tau_n^i + \sqrt{2(1-\lambda)\tau_n}\xi_n$. ▷ Reverse-time process step
    for $\ell = 0, 1, \dots, L$ **do**
      Sample $\xi_k \sim \mathcal{N}(0, I_d)$
      $z_{n+1}^i \longleftarrow z_{n+1}^i - \tau\nabla v_{\boldsymbol{Y}_{T-t_n}}(z_{n+1}^i) + \sqrt{2\tau}\xi_k$. ▷ Langevin post-processing step
    end for
  end for
end for

5.1 Verification result: Gaussian setting

Problem definition Let $d = 10$, $K = [-5, 5]^{10}$ and $\Phi(x) = x^\intercal \Sigma^{-1} x$, where Σ is a randomly generated symmetric positive definite matrix (we sample entries of a matrix A uniformly on $[0, 1]$ and then define $\Sigma^{-1} = A^\intercal A + 0.1 I_d$). Note that in this setting the polynomial degree of the HJB solution is bounded by $\boldsymbol{n} = (2, \dots, 2)$ as π_t remains a Gaussian density if π_0 and π_∞ are Gaussian.

Parameters For Algorithm 2, we choose $T = 12$, $\tau_{\max} = 0.1$, $\rho = 0.2$, $\delta_{\text{proj}} = \delta_{\text{rank}} = 0.01$, $\delta_{\text{contr}} = 10^{-8}$.

Evaluation By Lemma D.1, Φ has FTT- $\boldsymbol{r} = (3, 4, 5, 6, 7, 6, 5, 4, 3)$. Since the solution of the HJB is a strictly quadratic polynomial for all times (meaning that no higher or lower degrees than 2 appear), Lemma D.1 also yields that the FTT rank of the solution is bounded from above by $\boldsymbol{r}$ for all t_n. In Figure 5.1 the ranks of the solution during integration are displayed. Once the solution reaches a covariance error of $\sim 10^{-7}$, the solver starts to truncate the ranks, meaning that at this point higher ranks give a contribution to the solution which is less than $\delta_{\text{contr}} = 10^{-8}$ in relative Frobenius norm. Finally, all ranks higher than 2 are truncated, which is to be expected since the standard normal potential has FTT rank $\boldsymbol{r} \equiv 2$. At $t = 12$, the covariance error has decreased to $\sim 10^{-11}$.
Figure 5.2 displays the stepsizes chosen by the solution algorithm. Since the polynomial degree does not increase and the ranks are bounded from above by the initial rank, the stiffness estimate (4.11) determines the stepsize.

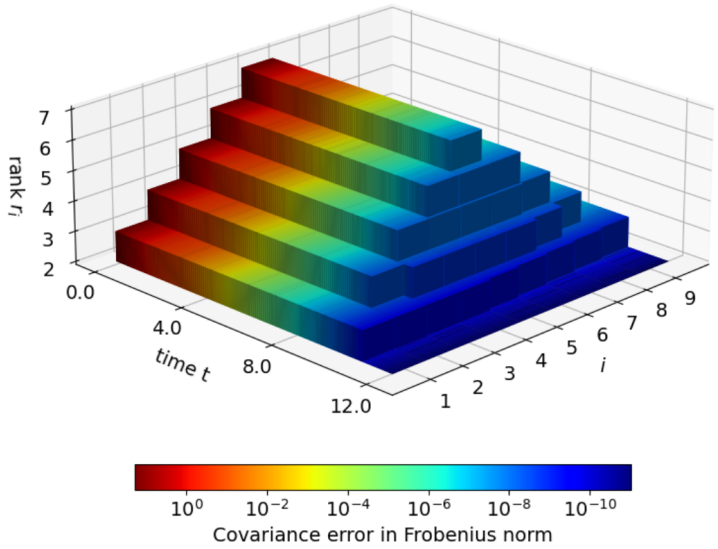

Figure 5.1: Development of the solution ranks and the covariance error (5.3) over time in the Gaussian setting. Once the solution is close to convergence (in terms of the covariance error), the ranks decrease to the rank $(2, \dots, 2)$ of the potential of the standard normal distribution.

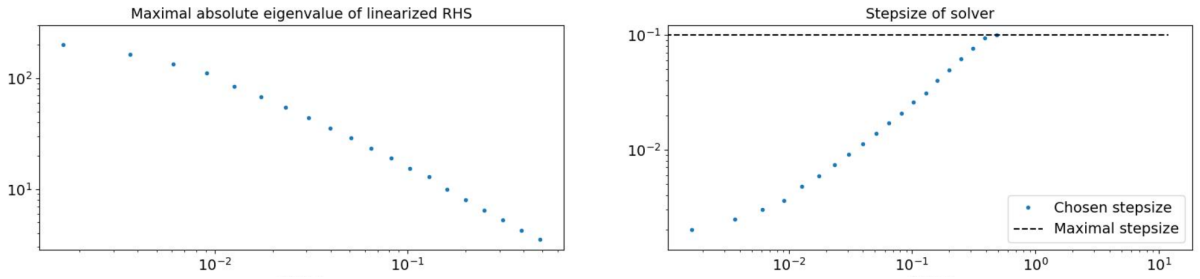

Figure 5.2: Approximations of the maximal absolute eigenvalues of the linearized right-hand side $|\overline{\lambda}_t|$ determined by the power method (left) and accordingly chosen stepsize $2\rho/|\overline{\lambda}_t|$ (right) over time in the Gaussian setting. Note that the eigenvalues decrease monotonically, permitting a monotonous increase of the stepsize until the maximal permitted stepsize $\tau_{\max} = 0.1$ is reached.

5.2 Mixed nonlinear density

Problem definition Let $d = 20$, $K = [-5,5]^2 \times [-2,2]^2 \times [-5,5]^2 \times [-2,2]^{14}$. Consider the transport map $\mathcal{T}\colon \mathbb{R}^2 \to \mathbb{R}^2$ and matrix Σ with

$$\mathcal{T}(x,y) = (x, y + x^2 + 1), \quad \Sigma = \begin{pmatrix} 1 & 0.9 \\ 0.9 & 1 \end{pmatrix}. \tag{5.4}$$

Let $\Phi_1(x,y) = v_\infty(\Sigma^{-1}\mathcal{T}(x,y))$, $\Phi_2(x,y) = x^4 + y^4 - 4x^2 - 4y^2 - 0.4x + 0.1y + 8$ and $\Phi_3(x,y) = x^6 + y^6 + 3xy$. Define $\Phi(x) = \Phi_1(x_1,x_2) + \Phi_2(x_3,x_4) + \Phi_3(x_5,x_6) + \sum_{i=7}^{20} x_i^2$. The first six dimensions of this potential define a *banana*-shaped marginal density in the first two dimensions, a nonsymmetric multimodal marginal density in the third and forth dimensions, and a bimodal marginal density in the fifth and sixth dimensions (see the right most column in Figure 5.3). By construction, this potential has rank $\boldsymbol{r} = (3,2,2,2,3,2,\dots,2)$.

Parameters We choose $\boldsymbol{n} = (4,2,4,4,6,6,2,\dots,2) \in \mathbb{N}^{20}$ according to the degrees appearing in the potential. For Algorithm 2 we set $T = 10$, $\tau_{\max} = 0.05$, $\delta_{\text{proj}} = \delta_{\text{rank}} = 0.01$, $\delta_{\text{contr}} = 10^{-8}$. To account for the high stiffness of the equation at small time $t \ll 1$, we set the stiffness parameter ρ in Algorithm 2 to $\rho = 0.001$ as long as $t < 10^{-6}$ and $\rho = 0.5$ for $t \geq 10^{-6}$. Langevin postprocessing (see Algorithm 3) is performed with $L = 100$ steps and stepsize $\tau = 0.005$.

Evaluation While the rank between independent parts of the density does not increase under the HJB flow, the initial ranks $r_1 = r_5 = 3$ may increase due to the time stepping scheme and hence incur a truncation error. However, with the specified values for ρ we discover that the stepsize resulting from the stiffness criterion (4.11) satisfies both the projection and the truncation criterion (4.13), (4.14) with the requested tolerance, suggesting that the solver keeps these errors sufficiently small. Figure 5.4 shows these stepsizes with a jump around $t = 10^{-6}$ due to the increase in the stiffness control parameter ρ. Figure 5.5 shows the exponential decay in the covariance error (5.3) between the HJB solution and the standard normal distribution. Note that there is an initial spike in the error for small times t. In experimentation, this spike seems to decrease in magnitude when permitting higher polynomial degrees. Hence, we can attribute it to a discretization error. The optimal choice of permitted degrees to balance accuracy and computational feasibility is an open question at this point. We conjecture that it is at this point that future research will prove most fruitful: the difficult region close to $t = 0$, where the true solution of the HJB is far away from the standard normal potential. Finally, Figure 5.3 shows the densities corresponding to the HJB solution obtained by Algorithm 2 and the samples at the corresponding time points in the reverse process. We note that the curvature of the banana potential in the first two dimensions as well as the multimodalities in higher dimensions are recovered by the method. Finally, we note the large number of postprocessing steps used in this example. We observed a drastic decrease in sample quality for less postprocessing steps.

6 Conclusion and Outlook

We presented an interpretable solver for the HJB equation arising from Hopf-Cole transformation of the Fokker-Planck equation in the setting of Bayesian inference and Generative Modelling. The approach uses functional Tensor Trains and spatial discretization with Legendre polynomials. A surro-

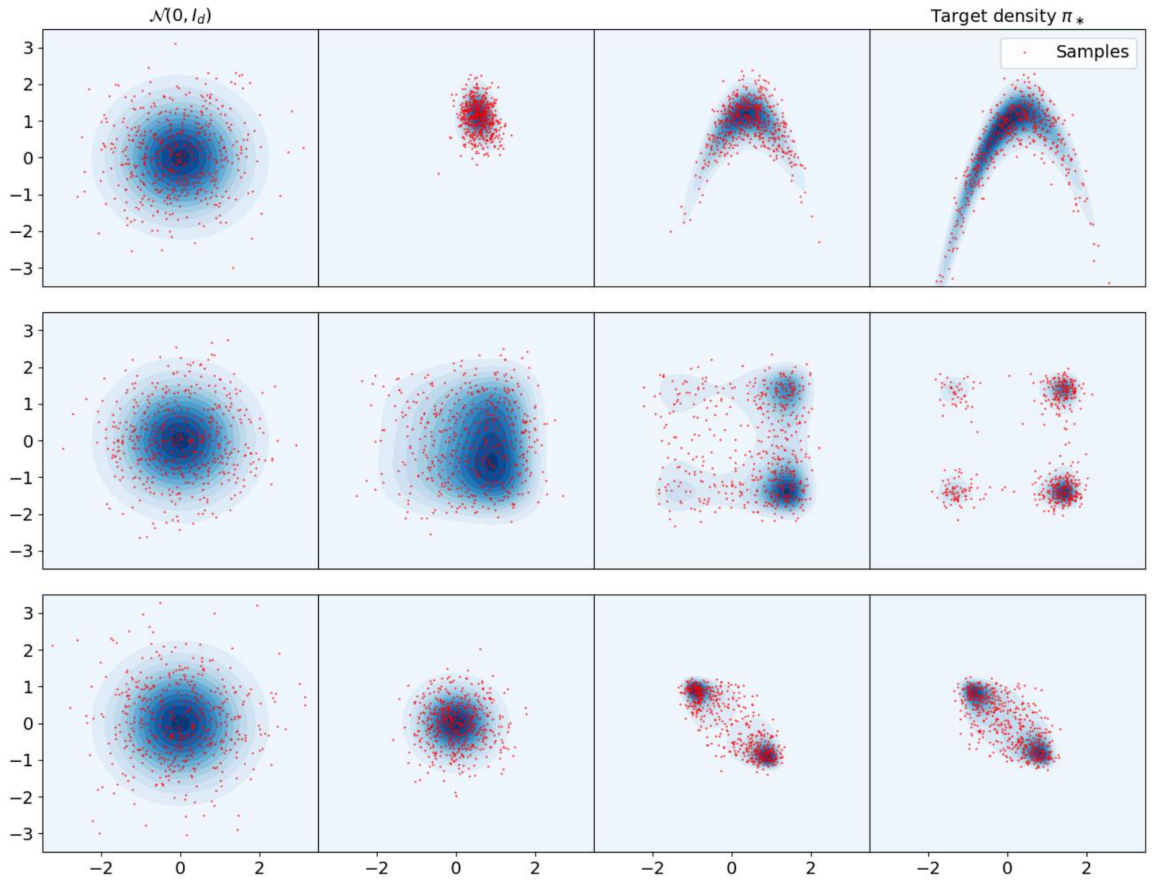

Figure 5.3: Development of marginal densities (blue) and the samples produced by the corresponding reverse process defined by Algorithm 3 (red) in the setting of the mixed nonlinear case (Section 5.2). The first row shows the values of the densities and samples on the (x_1, x_2)-plane, which is governed by the the Banana potential. The second row concerns the (x_3, x_4)-plane, which is governed by the nonsymmetric multimodal potential. The third row shows the (x_5, x_6)-dimension, governed by the bimodal potential. On the level of the HJB solver, the plot should be viewed *from right to left* since the target density (right) is transformed to a standard Gaussian (left). On the level of the reverse process, the samples (red) move from the standard Gaussian on the left to the target measure on the right. We note that in all cases the sampler is able to reproduce the multimodality and curvature of the corresponding density.

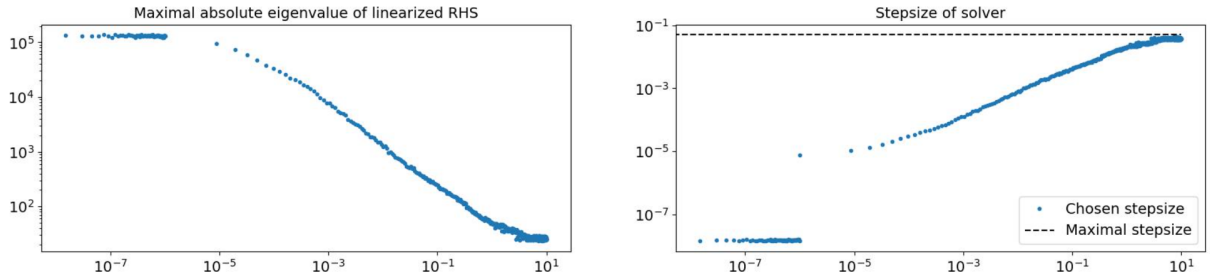

Figure 5.4: Approximations of the maximal absolute eigenvalues of the linearized HJB right-hand side (left) and accordingly chosen time stepsizes (right) as in Figure 5.2 but for the mixed nonlinear potential from Section 5.2. Note the jump in the stepsize at $t = 10^{-6}$ which corresponds to a change in the stiffnes control parameter ρ. Up to small perturbations which may be attributed to inaccuracy of the power method the stepsizes are monotonically increasing again.

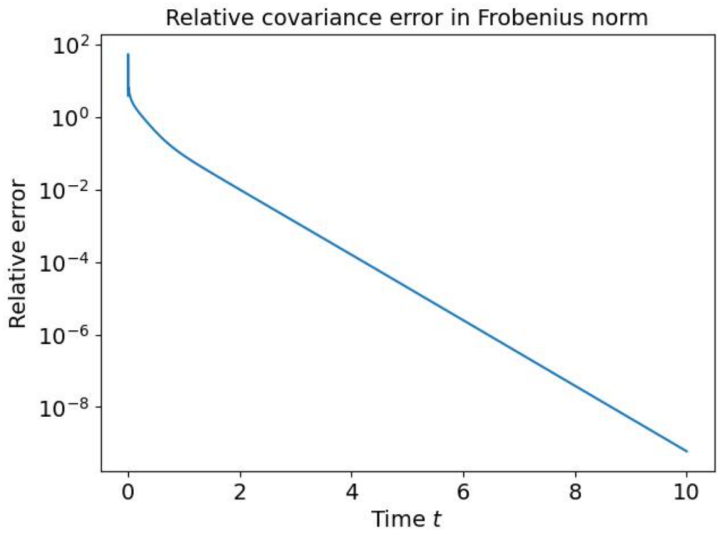

Figure 5.5: Decay of the covariance error (5.3) for the mixed nonlinear potential of Section 5.2. Note that after an initial spike, which may be attributed to degree and rank increase of the true HJB solution, the error decays exponentially.

gate replacement for the HJB equation, which reduces to an ODE on tensor space, was derived. The applicability of the method was demonstrated on linear and nonlinear test cases.

There are some obvious avenues for future work.

- Incorporating more recent state-of-the-art dynamical low rank integrators for matrix valued ODEs such as [8, 9] to the Tensor Train setting could lead to substantial performance improvements of the proposed method. In particular, the Basis Update $\&$ Galerkin (BUG) integrator [8] introduces rank adaptivity, while the fully parallel integrator [9] could additionally greatly speed up computations in high dimensions.

- Sampling from the reverse process via an Euler-Maruyama discretization usually requires a small step size and a high number of time steps. In a recent work [43], a *Diffusion Exponential Integrator Sampler* (DEIS) was proposed, which utilizes the semilinear structure of the learned diffusion process (2.4) to reduce the discretization error. This integrator could be applied in our setting. In particular, the combination with recent dynamical low rank solvers such as [9] could lead to a greatly reduced number of necessary steps both in solving the HJB as well as in discretizing the reverse process.

- We provided results for the FTT rank structure of the HJB solution in case of Gaussian distributions (Lemma D.1) and distributions with independent components (Lemma D.2) but it is an open question if there are further rank structures that are preserved under the HJB flow. As a fist step, Lemma D.2 can be generalized to independence between groups of components: Let $f(x) = f_1(x_1, \dots, x_n) + f_2(x_{n+1}, \dots, x_d)$. Then the FTT ranks of both f and $\mathrm{Lin}(f) + \mathrm{NonLin}(f)$ satisfy $r_n \leq 2$. We conjecture that there are further situations in which the solution ranks can be bounded:

  Conjecture: The HJB flow FTT ranks $\boldsymbol{r}_t$ are (up to a constant) bounded by $\boldsymbol{r}_0$ of v_0 and $\boldsymbol{r}_\infty \equiv 2$ for v_∞.

  This analysis is part of investigations in a subsequent work.

- Finally, a rigorous analysis providing error estimates between the solution of the projected equation (4.1) and the solution of the HJB equation (2.9) needs to be carried out. For a Gaussian potential, the solution of (4.1) coincides with that of (2.9). For more general densities the quality of the approxmiation largely depends on the initial condition, the contraction properties of the right-hand side of the HJB equation and the projection error.

References

[1] B. D. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

[2] M. Bachmayr. Low-rank tensor methods for partial differential equations. *Acta Numerica*, 32:1–121, 2023.

[3] M. Bachmayr, A. Nouy, and R. Schneider. Approximation by tree tensor networks in high dimensions: Sobolev and compositional functions. *arXiv preprint arXiv:2112.01474*, 2021.

[4] J. Berner, M. Dablander, and P. Grohs. Numerically solving parametric families of high-dimensional kolmogorov partial differential equations via deep learning. *Advances in Neural Information Processing Systems*, 33:16615–16627, 2020.

[5] J. Berner, L. Richter, and K. Ullrich. An optimal control perspective on diffusion-based generative modeling. *arXiv preprint arXiv:2211.01364*, 2022.

[6] S. Brooks, A. Gelman, G. Jones, and X.-L. Meng. *Handbook of Markov Chain Monte Carlo*. CRC press, 2011.

[7] E. Calvello, S. Reich, and A. M. Stuart. Ensemble Kalman methods: A mean field perspective, 2022.

[8] G. Ceruti, J. Kusch, and C. Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(4):1149–1174, 2022.

[9] G. Ceruti, J. Kusch, and C. Lubich. A parallel rank-adaptive integrator for dynamical low-rank approximation, 2023.

[10] A. Cohen, R. Devore, and C. Schwab. Analytic regularity and polynomial approximation of parametric and stochastic elliptic pde's. *Analysis and Applications*, 9(01):11–47, 2011.

[11] S. Dolgov, D. Kalise, and K. K. Kunisch. Tensor decomposition methods for high-dimensional hamilton–jacobi–bellman equations. *SIAM Journal on Scientific Computing*, 43(3):A1625–A1650, 2021.

[12] M. Eigel, R. Gruhlke, and D. Sommer. Less interaction with forward models in langevin dynamics, 2022.

[13] M. Eigel, R. Schneider, and D. Sommer. Dynamical low-rank approximations of solutions to the hamilton–jacobi–bellman equation. *Numerical Linear Algebra with Applications*, 30(3):e2463, 2023.

[14] A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart. Interacting Langevin diffusions: Gradient structure and ensemble Kalman sampler. *SIAM Journal on Applied Dynamical Systems*, 19(1):412–441, 2020.

[15] A. Garbuno-Inigo, N. Nusken, and S. Reich. Affine invariant interacting Langevin dynamics for Bayesian inference. *SIAM Journal on Applied Dynamical Systems*, 19(3):1633–1658, 2020.

[16] J. Goodman and J. Weare. Ensemble samplers with affine invariance. *Communications in applied mathematics and computational science*, 5(1):65–80, 2010.

[17] M. Griebel, H. Harbrecht, and R. Schneider. Low-rank approximation of continuous functions in sobolev spaces with dominating mixed smoothness. *Mathematics of Computation*, 92(342):1729–1746, 2023.

[18] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[19] S. Holtz, T. Rohwedder, and R. Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

[20] S. Holtz, T. Rohwedder, and R. Schneider. On manifolds of tensors of fixed tt-rank. *Numerische Mathematik*, 120(4):701–731, 2012.

[21] C.-W. Huang, J. H. Lim, and A. C. Courville. A variational perspective on diffusion-based generative models and score matching. *Advances in Neural Information Processing Systems*, 34:22863–22876, 2021.

[22] A. Hyvärinen and P. Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[23] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

[24] Q. Liu and D. Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[25] C. Lubich and I. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54:171–188, 2013.

[26] C. Lubich, I. V. Oseledets, and B. Vandereycken. Time integration of tensor trains. *SIAM Journal on Numerical Analysis*, 53(2):917–941, 2015.

[27] P. A. Markowich and C. Villani. On the trend to equilibrium for the fokker-planck equation: an interplay between physics and functional analysis. *Mat. Contemp*, 19:1–29, 2000.

[28] N. Nüsken and L. Richter. Solving high-dimensional hamilton–jacobi–bellman pdes using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2:1–48, 2021.

[29] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[30] I. V. Oseledets. Constructive representation of functions in low-rank tensor formats. *Constructive Approximation*, 37:1–18, 2013.

[31] M. Oster, L. Sallandt, and R. Schneider. Approximating optimal feedback controllers of finite horizon control problems using hierarchical tensor formats. *SIAM Journal on Scientific Computing*, 44(3):B746–B770, 2022.

[32] S. Reich and S. Weissmann. Fokker–Planck particle systems for Bayesian inference: Computational approaches. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):446–482, 2021.

[33] L. Rey-Bellet and K. Spiliopoulos. Improving the convergence of reversible samplers. *Journal of Statistical Physics*, 164:472–494, 2016.

[34] C. Robert and G. Casella. A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data. *Statistical Science*, 26(1):102 – 115, 2011.

[35] G. O. Roberts and J. S. Rosenthal. General state space Markov chains and MCMC algorithms. *Probability surveys*, 1:20–71, 2004.

[36] G. O. Roberts and R. L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, pages 341–363, 1996.

[37] Y. Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[38] Y. Song, S. Garg, J. Shi, and S. Ermon. Sliced score matching: A scalable approach to density and score estimation. In R. P. Adams and V. Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 574–584. PMLR, 22–25 Jul 2020.

[39] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[40] A. M. Stuart. Inverse problems: A bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[41] B. J. Zhang, Y. M. Marzouk, and K. Spiliopoulos. Transport map unadjusted langevin algorithms, 2023.

[42] K. S. Zhang, G. Peyré, J. Fadili, and M. Pereyra. Wasserstein control of mirror langevin monte carlo. In *Conference on Learning Theory*, pages 3814–3841. PMLR, 2020.

[43] Q. Zhang and Y. Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023.

[44] M. Zhou, J. Han, and J. Lu. Actor-critic method for high dimensional static hamilton–jacobi–bellman partial differential equations based on neural networks. *SIAM Journal on Scientific Computing*, 43(6):A4043–A4066, 2021.

A Hopf-Cole Transformation

Let π_t satisfy the Fokker-Planck equation

$$\partial_t \pi_t = \Delta \pi_t + x \cdot \nabla \pi_t + d\pi_t. \tag{A.1}$$

Then, by the chain and product rule, we have the identities

$$\partial_t \log \pi_t = \frac{1}{\pi_t} \partial_t \pi_t = \frac{1}{\pi_t}(\Delta \pi_t + x \cdot \nabla \pi_t + d\pi_t), \tag{A.2}$$

$$\Delta \log \pi_t = \nabla \cdot \left(\frac{1}{\pi_t} \nabla \pi_t \right) = \frac{1}{\pi_t} \Delta \pi_t - \frac{1}{\pi_t^2} \nabla \pi_t \cdot \nabla \pi_t = \frac{1}{\pi_t} \Delta \pi_t - \|\nabla \log \pi_t\|_2^2. \tag{A.3}$$

Putting these together, we see that

$$\partial_t \log \pi_t = \Delta \log \pi_t + \|\nabla \log \pi_t\|_2^2 + x \cdot \nabla \log \pi_t + d \tag{A.4}$$

and hence $v_t = -\log \pi_t$ satisfies

$$\partial_t v_t = \Delta v_t - \|\nabla v_t\|_2^2 + x \cdot \nabla v_t - d. \tag{A.5}$$

B Optimal Control Perspective

We consider the case where π_* is a zero-mean Gaussian with symmetric positive definite covariance matrix Σ. Hence, (2.9) becomes

$$\partial_t v_t = \mathrm{Lin}(v_t) + \mathrm{NonLin}(v_t), \qquad v_0 = \frac{1}{2} x^\intercal \Sigma^{-1} x. \tag{B.1}$$

This is a familiar form in stochastic optimal control. Consider an SDE

$$\mathrm{d}X_t = (f(X_t) + g(X_t)u_t)dt + \sigma(X_t)\mathrm{d}W_t, \tag{B.2}$$

$$X_0 = x_0, \tag{B.3}$$

with initial condition x_0, control u_t, diffusion σ, free drift part f and controlled drift part $g \cdot u_t$. Associate with this SDE a cost functional given by

$$J(t, x, u) = \mathbb{E}\left[\int_t^T \lambda u_t^\intercal u_t \mathrm{d}t + E(X_T) \middle| X_t = x \right], \tag{B.4}$$

where $\lambda > 0$ and E is a positive definite terminal cost function. The associated HJB equation for the value function V reads

$$\partial_t V + f^\intercal \nabla V - \frac{1}{4\lambda} \nabla V^\intercal g g^\intercal \nabla V + \frac{\sigma^2}{2} \Delta V = 0, \qquad V(T, x) = E(x).$$

Now, choose $f(x) = x$, $g(x) \equiv I_d$, $E(x) = x^\intercal \Sigma^{-1} x/2$, $\lambda = 1/4$ and $\sigma = \sqrt{2}$ to arrive at

$$\begin{aligned} \partial_t V &= -\nabla V^\intercal x + \|\nabla V\|^2 - \Delta V, \\ V(T, x) &= \frac{1}{2} x^\intercal \Sigma^{-1} x. \end{aligned} \tag{B.5}$$

Clearly, reversing the time by defining $\overleftarrow{V}(t,x) \coloneqq V(T-t,x)$ and regrouping the terms yields

$$\begin{aligned} \partial_t \overleftarrow{V} &= \Delta \overleftarrow{V} + \nabla \overleftarrow{V}^\intercal x - \|\nabla \overleftarrow{V}\|^2 = \mathrm{Lin}(\overleftarrow{V}) + \mathrm{NonLin}(\overleftarrow{V}), \\ \overleftarrow{V}(0,x) &= \frac{1}{2} x^\intercal \Sigma^{-1} x, \end{aligned} \tag{B.6}$$

and hence $v_t = \overleftarrow{V}(t,\cdot)$. In total, the log-density of the foward SDE is given as the reverse-time value function of a linear quadratic optimal control problem. The linear quadratic problem has the property that its solution is the same in the deterministic and stochastic setting. Hence, instead of the stochastic problem, we may also consider the deterministic optimal control problem defined by

$$\dot{x} = x + u, \qquad x(0) = x_0, \tag{B.7}$$

$$J(t,x,u) = \int_t^T \frac{1}{4} u(t)^\intercal u(t) \mathrm{d}t + \frac{1}{2} X(T)^\intercal \Sigma^{-1} X(T). \tag{B.8}$$

Now, setting $A = I_d$, $B = I_d$, $Q \equiv 0 \in \mathbb{R}^{d\times d}$, $R = \frac{1}{4} I_d$ and $Q_f = \frac{1}{2}\Sigma_\rho^{-1}$ this leads to

$$\dot{x} = Ax + Bu, \qquad x(0) = x_0, \tag{B.9}$$

$$J(t,x,u) = \int_t^T x(t) Q x(t) + u(t)^\intercal R u(t) \mathrm{d}t + X(T)^\intercal Q_f X(T). \tag{B.10}$$

The solution of this problem is given by the LQR $V(t,x) = x^\intercal P_t x$, where P_t solves a Riccati differential equation with inputs A, B, Q, R, Q_f. It follows that

$$\nabla v_{T-t}(x) = \nabla V(t,x) = 2 P_t x,$$

leading for $\lambda = 0$ to a reverse-time generative process defined by

$$\mathrm{d}Y_s = (I_d - 4P_s) Y_s \mathrm{d}s + \sqrt{2}\mathrm{d}W_s. \tag{B.11}$$

C Motivation for using (Functional) Tensor Trains

We give an informal motivation for the use of FTTs and in particular polynomials represented in the Tensor Train format in the setting of Bayesian inference for paramteric PDEs. For more rigorous representations and decay rates in polynomial chaos representations of solutions of parametric PDEs we refer e.g. to pioneering work in [10] and follow up research. In this setting, the fact that usually only very few data points are available often renders high frequency components non-informative. Hence, the higher mode dimensions are often close to the prior information even after inference. Assuming that the prior is given as a (standard) Gaussian, it is reasonable to assume that these higher modes will be close to Gaussian. In particular, this motivates a form of potential similar to the nonlinear potential used in Section 5.2. The following provides a sketch on how such a form might be obtained. Let $M_1 \in \mathbb{N}$, $M_1 < d$, denote a number of relevant modes and for maximal polynomial degrees $d_1 \geq d_2 \geq \ldots \geq d_{M_1} \geq 2$ let

$$\text{relevant} = \bigtimes_{i=1}^{M_1} \{0,\ldots,d_i\}, \qquad \sqrt{\text{relevant}} = \bigtimes_{i=1}^{M_1} \{0,\ldots,\lfloor\sqrt{d_i}\rfloor\}.$$

Observe that solutions u of parameteric PDEs with spatial variable x and parameter y can often be written as

$$u(x,y) \approx \sum_{\boldsymbol{\alpha}\in\sqrt{\text{relevant}}} u_{\boldsymbol{\alpha}}(x)p_{\boldsymbol{\alpha}}(y)$$

where $u_{\boldsymbol{\alpha}}$ is an element of some function space V for every $\boldsymbol{\alpha}$. Let $G(y) = u(\cdot,y) \in V$ and for some $K \in \mathbb{N}$ let $\mathcal{O}\colon V \to \mathbb{R}^K$ be a linear observation operator (e.g. point evaluations in x). Hence:

$$\mathcal{O}(G(y)) = \mathcal{O}(u(\cdot,y)) = \sum_{\boldsymbol{\alpha}\in\sqrt{\text{relevant}}} \boldsymbol{u}_{\boldsymbol{\alpha}}p_{\boldsymbol{\alpha}}(y), \quad \boldsymbol{u}_{\boldsymbol{\alpha}} \in \mathbb{R}^K.$$

Then, assuming a Bayesian setting with a zero mean Gaussian prior with covariance matrix Σ, the log posterior density has the form

$$\log \pi(y) = -\frac{1}{2}\|\mathcal{O}(G(y)) - \delta\|^2_{\sigma I_d} - \frac{1}{2}\|y\|^2_\Sigma$$

where δ is an observation and σI_d, $\sigma > 0$, is the covariance of the zero mean Gaussian observational noise. By the form of $\mathcal{O}(G(y))$, it then follows that there are coefficient tensors c^{prior} and $c^{\text{likelihood}}$ such that the potential or negative log posterior density is of the form with non-Gaussian parts confined to the relevant modes $1,\dots,M_1$.

D Functional Tensor Train rank of HJB solutions

Lemma D.1 (Gaussian distributions)**.** *Let $d \in \mathbf{N}$ and $f\colon \mathbb{R}^d \to \mathbb{R}$ admit the form $f(x) = x^\intercal Mx$ for a symmetric positive definite matrix $M \in \mathbb{R}^{d,d}$. Then f has finite FTT rank $\boldsymbol{r} \in \mathbb{N}^{d-1}$. In particular for $d \geq 3$,*

$$\boldsymbol{r} \leq \overline{\boldsymbol{r}} := 2 + \begin{cases} \left(1,2,\dots,\frac{d}{2},\dots,2,1\right), & d \text{ even}, \\ \left(1,2,\dots,\frac{d-1}{2},\frac{d-1}{2},\dots,2,1\right), & d \text{ odd}, \end{cases}$$

and $\boldsymbol{r} = 2 \in \mathbb{N}$ for $d = 2$.

Proof. The case $d = 2$ follows since M is invertible and the TT rank coincides with the matrix rank. Let $d \geq 3$ and write $M = (m_{ij})$ and $\overline{\boldsymbol{r}} = (\overline{r}_i)_{i=1}^{d-1}$, $\overline{r}_0 = \overline{r}_d = 1$. We seek a representation

$$f(x) = U_1(x_1)U_2(x_2)\cdots U_d(x_d), \quad U_i(x_i) \in \mathbb{R}^{r_{i-1},r_i}, i = 1,\dots,d.$$

Let $I_n \in \mathbb{R}^{n,n}$ denote the identity matrix and $\mathbf{0}_{k,l} \in \mathbb{R}^{kl}$ be a zero matrix and $\mathbf{0}_k \in \mathbb{R}^k$ be a zero vector.

Define the matrices $\tilde{U}_i(x_i)$ for $i = 1, d$ as

$$\tilde{U}_1(x_1) = \begin{pmatrix} 1 & 2x_1 & m_{11}x_1^2 \end{pmatrix} \in \mathbb{R}^{1,\overline{r}_1}, \qquad \tilde{U}_d(x_d) = \begin{pmatrix} 1 & 2x_d & m_{dd}x_d^2 \end{pmatrix}^\intercal \in \mathbb{R}^{\overline{r}_{d-1},1}.$$

Moreover, for $i = 2,\dots,d-1$ except for $i = \frac{d-1}{2}+1$ in case hat d is odd let

$$\tilde{U}_i(x_i) = \left[\begin{array}{c|c|c} & 2x_i & m_{ii}x_i^2 \\ & & m_{1i}x_i \\ I_i & \mathbf{0}_{i-1} & \vdots \\ & & m_{i-1,i}x_i \\ \hline \mathbf{0}_i^\intercal & 0 & 1 \end{array}\right] \in \mathbb{R}^{\overline{r}_{i-1},\overline{r}_i}, \quad i = 2,\dots,\left\lceil\frac{d-1}{2}\right\rceil,$$

$$\tilde{U}_i(x_i) = \left[\begin{array}{c|c|c} \begin{matrix}1\\ 2x_i\end{matrix} & \mathbf{0}_{2,d-i} & \mathbf{0}_2 \\ \hline \mathbf{0}_{d-i} & I_{d-i} & \mathbf{0}_{d-i} \\ \hline m_{i,i}x_i^2 & \begin{matrix}m_{d,i}x_i & \cdots & m_{i+1,i}x_i\end{matrix} & 1 \end{array}\right] \in \mathbb{R}^{\overline{r}_{i-1},\overline{r}_i} \quad i = \left\lfloor \frac{d+3}{2} \right\rfloor, \dots, d-1.$$

If d is odd we define the middle square component $U_i(x_i)$ for $i = \frac{d-1}{2} + 1$ by

$$\tilde{U}_i(x_i) = \left[\begin{array}{c|c|c} m_{ii}x_i^2 & \begin{matrix}m_{i,i+1}x_i & \cdots & m_{i,d}x_i\end{matrix} & 1 \\ \hline \begin{matrix}m_{1,i}x_i\\ \vdots \\ m_{i-1,i}x_i\end{matrix} & \tfrac{1}{2}M_{1:i-1,i+1:d} & \mathbf{0}_{i-1} \\ \hline 1 & \mathbf{0}_{d-i}^\intercal & 0 \end{array}\right], \quad i = \frac{d-1}{2} + 1.$$

For d even we define for $i = \frac{d}{2} + 1 = \left\lfloor \frac{d+3}{2} \right\rfloor$

$$U_i(x_i) = \left[\begin{array}{c|c|c} 0 & \mathbf{0}_{d-i}^\intercal & 1 \\ \hline \mathbf{0}_{i-1} & M_{1:i-1,i+1:d} & \mathbf{0}_{i-1} \\ \hline 1 & \mathbf{0}_{d-i}^\intercal & 0 \end{array}\right] \tilde{U}(x_i)$$

and in any other case set $U_i(x_i) = \tilde{U}_i(x_i)$. □

Lemma D.2 (Measures of independent variables)**.** *Let $d \in \mathbb{N}$ and $f\colon \mathbb{R}^d \to \mathbb{R}$ admit the form $f(x) = \sum_{i=1}^d f_i(x_i)$ for $f_i \in \mathcal{C}^2(\mathbb{R},\mathbb{R})$, $i = 1,\dots,d$. Then both f and $\mathrm{Lin}(f) + \mathrm{NonLin}(f)$ have FTT rank $\mathbf{r} = (2,\dots,2)^\intercal \in \mathbb{N}^{d-1}$.*

Proof. The result follows immediately from [30, Theorem 2] and the structure of $\mathrm{Lin}(f)+\mathrm{NonLin}(f)$. □

E Details of HJB solutions

Let $p_\alpha^{(\mathrm{mon})}$ for $\alpha \in \mathbb{N}_0$ denote the α-th monomial, i.e. $p_\alpha^{(\mathrm{mon})}(x) = x^\alpha$. As in Section 3, let $T_{i,n} \in \mathbb{R}^{n+1,n+1}$ denote the basis transformation matrix between Legendre polynomials of degree n on $[a_i, b_i]$ and the monomials up to degree n.

E.1 Derivation matrices in the linear operator part

Note that for every $i = 1,\dots,d$ and for every $c \in \mathbb{R}^{n_i+1}$, we have

$$\partial_x^2 \sum_{\alpha=0}^{n_i} c_\alpha p_\alpha^{(\mathrm{mon})} = \sum_{\alpha=0}^{n_i} (M_{dd}^i c)_\alpha p_\alpha^{(\mathrm{mon})}, \tag{E.1}$$

where

$$M_{dd}^i = \begin{pmatrix} 2 & & & & \\ & 6 & & & \\ & & \ddots & & \\ & & & n_i(n_i-1) & \\ & & & & 0 \\ & & & & 0 \end{pmatrix} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}. \tag{E.2}$$

In a similar way, we get

$$x\partial_x \sum_{\alpha=0}^{n_i} c_\alpha p_\alpha^{(\mathrm{mon})} = \sum_{\alpha=0}^{n_i} (M_{xd}^i c)_\alpha p_\alpha^{(\mathrm{mon})}, \tag{E.3}$$

where

$$M_{xd}^i = \begin{pmatrix} 0 & & & & \\ & 1 & & & \\ & & 2 & & \\ & & & \ddots & \\ & & & & n_i \end{pmatrix} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}. \tag{E.4}$$

With the basis transformation matrix T_{i,n_i} we can express the action of these operators on the coefficients of the original basis p_i. In particular, we have

$$\begin{aligned} (\partial_x^2 + x\partial_x) \sum_{\alpha=0}^{n_i} c_\alpha p_\alpha^i &= (\partial_x^2 + x\partial_x) \sum_{\alpha=0}^{n_i} (T_{i,n_i} c)_\alpha p_\alpha^{(\mathrm{mon})} \\ &= \sum_{\alpha=0}^{n_i} ((M_{dd}^i + M_{xd}^i) T_{i,n_i} c)_\alpha p_\alpha^{(\mathrm{mon})} \\ &= \sum_{\alpha=0}^{n_i} (T_{i,n_i}^{-1} (M_{dd}^i + M_{xd}^i) T_{i,n_i} c)_\alpha p_\alpha = \sum_{i=0}^{n_i} (D_i c)_\alpha p_\alpha^i, \end{aligned} \tag{E.5}$$

with $D_i \coloneqq T_{i,n_i}^{-1}(M_{dd}^i + M_{xd}^i)T_{i,n_i} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}$.

E.2 Derivation of the nonlinear part

Note that for every $c \in \mathbb{R}^{n_i+1}$, we have

$$\partial_x \sum_{\alpha=0}^{n_i} c_\alpha p_\alpha^{(\mathrm{mon})} = \sum_{\alpha=0}^{n_i} (M_d^i c)_\alpha p_\alpha^{(\mathrm{mon})}, \tag{E.6}$$

where

$$M_d^i = \begin{pmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & n_i \\ & & & 0 \end{pmatrix} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}, \tag{E.7}$$

and hence

$$\partial_x \sum_{\alpha=0}^{n_i} c_\alpha p_\alpha^i = \partial_x \sum_{\alpha=0}^{n_i} (T_{i,n_i} c)_\alpha p_\alpha^{(\mathrm{mon})} \tag{E.8}$$

$$= \sum_{\alpha=0}^{n_i} (M_d^i T_{i,n_i} c)_\alpha p_\alpha^{(\mathrm{mon})} \tag{E.9}$$

$$= \sum_{\alpha=0}^{n_i} (T_{i,n_i}^{-1} M_d^i T_{i,n_i} c)_\alpha p_\alpha^i = \sum_{\alpha=0}^{n_i} (D_{x_i} c)_\alpha p_\alpha^i, \tag{E.10}$$

with $D_{x_i} = T_{i,n_i}^{-1} M_d^i T_{i,n_i} \in \mathbb{R}^{(n_i+1)\times(n_i+1)}$.

E.3 Estimating the eigenvalues for Gaussian distributions

Let $\mathbf{n} = (2, \dots, 2)^\intercal \in \mathbb{N}^d$ and $g(x) = \frac{1}{2} x^\intercal A x$ for all $x \in \mathbb{R}^d$, where $A \in \mathbb{R}^{d\times d}$. Note that $\nabla g(x) = Ax$ and hence for any $v \in \operatorname{span} \Pi_{\mathbf{n}}$, we have $\mathrm{NonLin}_g\, v = \langle Ax, \nabla v\rangle = (Ax)\cdot\nabla v$. Hence, the linearized HJB at g reads

$$\dot v = x\cdot\nabla v + \Delta v - 2\langle Ax, \nabla v\rangle + \langle Ax, Ax\rangle, \qquad v(0) = g \tag{E.11}$$

In order to determine the stiffness, we need to determine the effect of this right-hand side on the coefficient tensor of v. From now on, let $v = v_{\boldsymbol{C}}$, where $\boldsymbol{C} \in \mathbb{R}^{\boldsymbol{n}+1}$. Assume that $v_{\boldsymbol{C}}(x) = \frac{1}{2} x^\intercal \hat C x$ for some $\hat C \in \mathbb{R}^{d\times d}$, i.e. $V_{\boldsymbol{C}}$ has only terms with degree 2. We know that $x\cdot\nabla v_{\boldsymbol{C}} + \Delta v_{\boldsymbol{C}} = v_{\boldsymbol{LC}}$ and $\langle Ax, \nabla v_{\boldsymbol{C}}\rangle = \sum_{i=1}^d \sum_{j=1}^d a_{ij} x_j \partial_i v_{\boldsymbol{C}}$. Now, note that $\partial_i v_{\boldsymbol{C}} = v_{(I_d\otimes\ldots\otimes P_i\otimes\ldots\otimes I_d)\boldsymbol{C}}$, where

$$P_i = T_{i,2}^{-1} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix} T_{i,2}, \tag{E.12}$$

and $x_j v_{\boldsymbol{C}} = v_{(I_d\otimes\ldots\otimes X_j\otimes\ldots\otimes I_d)\boldsymbol{C}}$, where

$$X_j = T_{j,2}^{-1} \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} T_{j,2}. \tag{E.13}$$

In the case of $i = j$ this leads to $x_i\partial_i V_C = v_{(I_d\otimes\ldots\otimes T_{i,2}^{-1} M_{xd}^i T_{i,2}^{-1}\otimes\ldots\otimes I_d)\boldsymbol{C}}$, where

$$M_{xd}^i = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}. \tag{E.14}$$

Let

$$\boldsymbol{M} = \sum_{i,j=1}^d a_{ij} I_d \otimes \ldots \otimes I_d \otimes P_i \otimes I_d \otimes \ldots \otimes I_d \otimes X_j \otimes I_d \otimes \ldots \otimes I_d, \tag{E.15}$$

then we have $\langle Ax, \nabla v_{\boldsymbol{C}}\rangle = v_{\boldsymbol{MC}}$.

Diagonal covariance. We consider the special case where $A = \mathrm{diag}(a_{ii}, i = 1, \dots, d)$ is a diagonal matrix. In this case, we have

$$\boldsymbol{M} = \sum_{i=1}^{d} a_{ii} I_d \otimes \dots I_d \otimes T_{i,2}^{-1} M_{xd}^i T_{i,2} \otimes I_d \otimes \dots \otimes I_d \tag{E.16}$$

and hence the linear operator governing the right-hand side is given by

$$\boldsymbol{L} - 2\boldsymbol{M} = \sum_{i=1}^{d} I_d \otimes \dots I_d \otimes H_i \otimes I_d \otimes \dots \otimes I_d, \tag{E.17}$$

where $H_i \coloneqq D_i - 2a_{ii} T_{i,2}^{-1} M_{xd}^i T_{i,2} = T_{i,2}^{-1}(M_{dd}^i + (1 - 2a_{ii}) M_{xd}^i) T_{i,2}$ and

$$M_{dd}^i + (1 - 2a_{ii}) M_{xd}^i = \begin{pmatrix} 0 & 0 & 2 \\ 0 & 1 - 2a_{ii} & 0 \\ 0 & 0 & 2(1 - 2a_{ii}). \end{pmatrix} \tag{E.18}$$

The point spectrum $\sigma(H_i)$ of H_i is given by $\sigma(H_i) = \{0, 1 - 2a_{ii}, 2(1 - 2a_{ii})\}$. The eigenvector corresponding to the eigenvalue with largest absolute value $2(1 - 2a_{ii})$ is given by

$$\hat{v}_i = T_{i,2}^{-1} \left(\frac{1}{1 - 2a_{ii}}, 0, 1 \right)^\intercal T_{i,2}. \tag{E.19}$$

Let v_i denote any eigenvector of H_i. Then, $v = (v_1 \otimes \dots \otimes v_d)$ is an eigenvector of $\boldsymbol{L} - 2\boldsymbol{M}$. Since this leads to 3^d possible combinations, the whole spectrum of $\boldsymbol{L} - \boldsymbol{M}$ is defined by such eigenvectors. Moreover, since the eigenvalues H_i are bounded by $|2(1 - 2a_{ii})|$, the largest absolute eigenvalue of $\boldsymbol{L} - 2\boldsymbol{M}$ is given by $2 \sum_{i=1}^{d} |1 - 2a_{ii}|$.