A class of mesh-free algorithms for finance, machine learning, and fluid dynamics

Philippe G. LeFloch* and Jean-Marc Mercier†

March 2021

Abstract

We introduce a numerical methodology which applies to a broad class of partial differential equations and discrete models, and is referred to here as the *transport-based mesh-free method*. It led us to several numerical algorithms which are now implemented in a Python library, called CodPy. We develop a mesh-free discretization technique based on the (so-called RKHS) theory of reproducing kernels and the theory of transport mappings, in a way that is reminiscent of Lagrangian methods in computational fluid dynamics. The strategy is relevant when a large number of dimensions or degrees of freedom are present, as is the case in mathematical finance and machine learning, but is also applicable in fluid dynamics. We present our algorithms primarily for the Fokker-Planck-Kolmogorov system of mathematical finance and for neural networks based on support vector machines. The proposed algorithms are nonlinear in nature and enjoy quantitative error estimates based on the notion of discrepancy error, which allow one to evaluate the relevance and accuracy of given data and numerical solutions.

Contents

1 Introduction

We consider several typical problems arising in financial engineering and data science and propose a numerical methodology which is mesh-free and based on a combination of analytical and algorithmic techniques, including the theory of reproducing kernels (also referred to as the RKSH theory) and the theory of transport mappings. By expanding our earlier work [14]–[17], the proposed *transport-based mesh-free method* applies to a broad class of partial differential equations (PDEs) and discrete problems and is particularly relevant when a large number of dimensions are involved.

The methodology advocated in the present paper is motivated by earlier work in fluid and material dynamics and in statistics. We refer the reader to [3] for background material on kernels and their application in computational statistics. For contributions in approximation theory, we refer to [2, 7, 8, 25, 31, 32, 36, 37], while for development of mesh-free methods in fluid dynamics and material sciences we refer to [4, 9, 12, 22, 23, 24, 27, 30, 38]. Designing

*Laboratoire Jacques-Louis Lions, Centre National de la Recherche Scientifique, Sorbonne Université, 4 Place Jussieu, 75252 Paris, France. Email: contact@philippelefloch.org.

†MPG-Partners, 136 Boulevard Haussmann, 75008 Paris, France. Email: jean-marc.mercier@mpg-partners.com.

algorithms that merge together techniques of machine learning and numerical PDEs is a very active domain of research [10, 11, 33].

Our approach provides a path to tacking the so-called curse of dimensionality which limits the performance of numerical algorithms in large dimensions. After introducing elementary material about the theory of reproducing kernels which serves as a backbone for the development of our numerical algorithms, we begin in Section 2 by presenting kernel-based discretization formulas for differential operators and extrapolation/interpolation operators. By using such operators, we can guarantee that our numerical algorithms are consistent with the problem under consideration, robust, and stable, as demonstrated by our numerical tests illustration. (The theoretical analysis of the algorithms is outside the scope of the present paper, which is focused on the presentation of the techniques and applications.)

We consider three domains of application of independent interest, as follows.

- Machine learning (Section 3): we revisit the formulation of Support Vector Machine arising in data science, and formulate an algorithm that combines together three levels and possibly several kernels.
- Mathematical finance (Section 4): we introduce a kernel-based discrete scheme for the approximation of the (forward) Fokker-Planck equation and the (backward) Kolmogorov equation, respectively.
- Polar decompositions (Section 5): we revisit the classical problem of the (signed) polar decomposition of mappings, and propose continuous and discrete algorithms.

For each problem we rely on our notions of kernel-based discretization, interpolation, and extrapolation. The discrete solutions depend upon the selection of a reproducing kernel and enjoy certain stability properties and are endowed with (a posteriori in nature) error estimates based on the notion of discrepancy distance and discrepancy sequences. We emphasize that our schemes are nonlinear in nature, even when the problems under consideration are linear. Each algorithm is illustrated by numerical experiments which demonstrate the efficiency of the proposed approach. While the present paper describes the algorithms and contains only basic numerical tests, while the companion papers [18]–[21] present the implementation of our algorithms as a Python code, called CodPy (standing for the "Curse of dimensionality in Python"). We refer the reader to this later paper for more specific problems and applications.

The use of the reproducing kernel techniques, which we advocate here and in [14]–[17], has several motivations. An important feature implied by the use of reproducing kernels is the possibility of computing error estimates based on the notion of discrepancy distance which allows for a *quantitative error evaluation* of the integration error or the discretization error associated with a given algorithm and a given set of data or solutions. This is a significant advantage in comparison to standard approaches used in in artificial neural networks and mathematical finance. Error estimates are essential in applications in order to decide the relevance of numerical output, and may also provide an important feedback on the relevance of the mathematical model itself. In addition, *kernel engineering techniques* (based on sums, products, or compositions of kernels) fit well within such a framework.

We build upon the results established earlier in [17] in which we systematically investigated, from the computational standpoint, kernel-based mesh-free approximations of multi-dimensional integrals. A kernel typically captures regularity and qualitative properties of functions "beyond" the standard Sobolev regularity class, which is an essential flexibility for many problems under consideration in data science and finance. In [17], we made comparisons between several numerical strategies, several choices of the kernel, and investigated the role played by the number of discretization points and the dimension of the problem. Quantitative error estimates in terms of a discrepancy distance are available for kernel-based algorithms, while such estimates are not available for purely random algorithms.

An outline of this paper is as follows. In Section 2 we present mesh-free discretization of integrals and differential operators, while Section 3 is devoted to their applications to support vector machines in machine learning. In Section 4, we present an algorithm for the Fokker-Plank-Kolmogorov system in mathematical finance, and finally Section 5 we are discuss an algorithm for the signed-polar decomposition in transport theory.

2 Mesh-free discretization of integrals and differential operators

2.1 Discrepancy and transport

Error integration estimate. We begin with an informal discussion restricted to the whole space $\mathbb{R}^D$ (with $D \geq 1$), before presenting our discretization formulas for differential operators and extrapolation/interpolation operators.

Introducing mesh-free, kernel-based discretization formulas is our main objective in this section, since they will be at the heart of the algorithms of the following three sections. Observe that sufficient regularity is assumed throughout this paper, so we do not discuss any issue on mathematical analysis.

Given an unstructured mesh in $\mathbb{R}^D$ represented by a set of N distinct points $Y = (y^1, \dots, y^N)$, we consider an approximation of any given probability measure μ on $\mathbb{R}^D$ by a sum of Dirac masses δ_{y^n} with equal weights, that is,

$$\mu \simeq \delta_Y = \frac{1}{N} \sum_{1 \le n \le N} \delta_{y^n}. \tag{2.1}$$

The accuracy of this approximation is determined by a distance among measures for averages of continuous and globally integrable functions φ. To this end, a functional space denoted by $\mathcal{H}^K(\mathbb{R}^D)$ is defined and, in this space, we seek an estimate of the form

$$\Big| \int_{\mathbb{R}^D} \varphi \, d\mu - \frac{1}{N} \sum_{1 \le n \le N} \varphi(y^n) \Big| \le \mathbf{d}^K(\mu, \delta_Y) \|\varphi\|_{\mathcal{H}^K(\mathbb{R}^D)}, \tag{2.2}$$

where, in the setup under consideration, the optimal coefficient $\mathbf{d}^K(\mu, \delta_Y)$ can be expressed explicitly in terms of the kernel and the set of points.

Discrepancy error. More precisely, $\mathcal{H}^K(\mathbb{R}^D)$ is the so-called reproducing kernel Hilbert space associated with a prescribed kernel K. By definition, the latter is a continuous and symmetric function $K : (x, y) \in \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}$ such that the matrix $K(Y, Y) = \big(K(y^n, y^m)\big)_{1 \le n,m \le N}$ is (symmetric) positive-definite for any set of distinct points. The Hilbert space of interest $\mathcal{H}^K(\mathbb{R}^D)$ defined by completion of $\mathrm{Span}\big\{K(\cdot, x) \,/\, x \in \mathbb{R}^D\big\}$ consists of all linear combinations of the functions $K(x, \cdot)$ (parametrized by $x \in \mathbb{R}^D$). This space is naturally endowed with a scalar product (see (2.10) below), and the coefficient $\mathbf{d}^K(\mu, \delta_Y)$ in (2.2) is referred to as the *discrepancy error* and reads

$$\mathbf{d}^K(\mu, \delta_Y)^2 = \iint_{\mathbb{R}^D \times \mathbb{R}^D} K(x, y) d\mu(x) d\mu(y) + \frac{1}{N^2} \sum_{n,m=1}^{N} K(y^n, y^m) - \frac{2}{N} \sum_{1 \le n \le N} \int_{\mathbb{R}^D} K(x, y^n) d\mu(x). \tag{2.3}$$

A set of points $\overline{Y}$ is called a *sharp discrepancy set* if it achieves the global minimum of the functional, that is, $\overline{Y} = \arg\inf_Y \mathbf{d}^K(\mu, \delta_Y)$. Even if minimum is not achieved exactly, in numerical applications we can seek a numerical approximation $\overline{Y}$ and use $\mathbf{d}^K(\mu, \delta_{\overline{Y}})$ as our discrepancy error in (2.2). This is the setup in which we can evaluate the error made in our algorithms below.

Methodology and transport maps. An additional feature of our approach is as follows. Given any discrete or continuous problem we can choose an admissible kernel K adapted the problem and introduce the corresponding functional space $\mathcal{H}^K(\mathbb{R}^D)$. We will then have various discrete approximation formulas as described in the rest of this section, including (2.1) whose accuracy is determined by computing the error function (2.3) in (2.2). Then, in order to optimize the convergence rate we can compute numerically an approximation of a sequence of points achieving (4.14). From a practical point of view, the discrepancy error can computed quite efficiently by, for instance, a direct Monte-Carlo approach as we do later in the test presented in this paper. We thus have a concrete approach for evaluating the accuracy of our approximation in *quantitative* way, which is essential in many applications.

In this juncture, we can also apply an optimal transport step [35], as follows. Let us consider a convex and open set $\Omega \subset \mathbb{R}^D$ with piecewise smooth boundary and, typically, we may take $(0, 1)^D$. Using a transportation argument, we reduce the study (2.2) to the same problem for the Lebesgue measure $\lambda = dx$ on Ω. Namely, consider the transport map $S : \Omega \mapsto \mathbb{R}^D$ associated with a given measure μ (with convex support), that is, the unique map satisfying

$$\int_{\mathbb{R}^D} \varphi d\mu = \int_{\Omega} (\varphi \circ S) dx, \tag{2.4}$$

for all continuous and globally integrable functions $\varphi : \mathbb{R}^D \to \mathbb{R}$. that moreover takes the form $S = \nabla h$ for some convex maps h (∇ denoting the gradient operator). The inequality (2.2) is then equivalent to

$$\Big| \int_{\Omega} (\varphi \circ S) \, dx - \frac{1}{N} \sum_{1 \le n \le N} (\varphi \circ S)(x^n) \Big| \le \mathbf{d}^K(\lambda, \delta_{S^{-1}Y}) \|\varphi \circ S\|_{\mathcal{H}^K(\Omega)}, \tag{2.5}$$

with $y^n = S(x^n)$, where $\varphi \circ S$ denotes the composition of two maps and $\mathcal{H}^K(\Omega)$ is the space defined on Ω (see below).

2.2 Notation for the continuous framework

Admissible kernels. Since we are primarily interested in kernels defined on a bounded set, we focus our presentation for now to this class —although we will allow ourselves to manipulate kernels defined on the whole Euclidian space. (For instance, such kernels may be used in order to generate the kernels of actual interest). A reproducing kernel [7, 36] provides a convenient way to generate a broad class of Hilbert spaces. A bounded and continuous function $K : \Omega \times \Omega \to \mathbb{R}$ on a bounded open set $\Omega \subset \mathbb{R}^D$ is called an *admissible kernel* if it satisfies (1) the *symmetry property:* $K(x, y) = K(y, x)$ for all $x, y \in \Omega$, and (2) the *positivity property:* for any collection of N distinct points $Y = (y^1, \dots, y^N)$ in Ω, the symmetric matrix $K(Y, Y) = \big(K(y^m, y^n)\big)_{1 \le n,m \le N}$ is positive definite in the sense that $a^T K(Y, Y)a > 0$ for all $a \in \mathbb{R}^N \setminus \{0\}$. It is said to be *uniformly positive* if there exists a uniform constant $c > 0$ such that for any collection of distinct points Y one has $a^T K(Y, Y)a \ge c\,|a|^2$ for all $a \in \mathbb{R}^N$.

Clearly, any admissible kernel satisfies

$$K(x, x) \ge 0, \qquad K(x, y)^2 \le K(x, x)\, K(y, y), \qquad x, y \in \Omega. \tag{2.6}$$

This implies that $2\, K(x, y) \le K(x, x) + K(y, y)$ and, therefore, the non-negative function

$$D(x, y) = K(x, x) + K(y, y) - 2K(x, y) \ge 0, \qquad x, y \in \Omega, \tag{2.7}$$

can be interpreted as a "pseudo-distance" in view of the properties $D(x, x) = 0$ and $D(x, y) = D(y, x)$. The triangle inequality need not hold; however the example $K_1(x, y) = 1 - |x - y|$ on $[0, 1]$ gives $D_1(x, y) = 2\,|x - y|$ which does satisfy the triangle inequality.

Functional spaces. Given an admissible kernel $K : \Omega \times \Omega \to \mathbb{R}$, we now introduce the (infinite dimensional) space $\widetilde{\mathcal{H}}^K(\Omega)$ consisting of all finite linear combinations of the functions $K(x, \cdot)$ parametrized by $x \in \Omega$, that is,

$$\widetilde{\mathcal{H}}^K(\Omega) = \mathrm{Span}\big\{K(\cdot, x) \,/\, x \in \Omega\big\}, \tag{2.8}$$

which we endow with the scalar product and norm defined as follows. To any two functions $\varphi = \sum_{1 \le m \le N} a_m K(\cdot, y^m)$ and $\psi = \sum_{1 \le n \le N} b_n K(\cdot, y^n)$ for any N distinct points $Y = (y^1, \dots, y^N)$ in Ω, we associate the bilinear expression (with $a = (a_m)$, etc.)

$$\langle \varphi, \psi \rangle_{\widetilde{\mathcal{H}}^K(\Omega)} = a^T K(Y, Y)b = \sum_{1 \le m \le N} \sum_{1 \le n \le N} a_m b_n K(y^m, y^n), \tag{2.9}$$

which endows the space with a Hilbertian structure with norm $\|\varphi\|^2_{\widetilde{\mathcal{H}}^K(\Omega)} = a^T K(Y, Y)a$. (Here, both Y and N may be arbitrary.) By construction, the reproducing kernel property holds in $\widetilde{\mathcal{H}}^K(\Omega)$:

$$\langle K(\cdot, x), K(\cdot, y) \rangle_{\widetilde{\mathcal{H}}^K(\Omega)} = K(x, y), \qquad x, y \in \Omega. \tag{2.10a}$$

From the Cauchy-Schwarz inequality, it follows that for any $\varphi \in \widetilde{\mathcal{H}}^K(\Omega)$ and $x \in \Omega$

$$|\varphi(x)| = |\langle K(\cdot, x), \varphi \rangle_{\widetilde{\mathcal{H}}^K(\Omega)}| \le \|K(\cdot, x)\|_{\widetilde{\mathcal{H}}^K(\Omega)}\, \|\varphi\|_{\widetilde{\mathcal{H}}^K(\Omega)} = \sqrt{K(x, x)}\, \|\varphi\|_{\widetilde{\mathcal{H}}^K(\Omega)}. \tag{2.10b}$$

Since the kernel is continuous and bounded, the "point evaluation" $\varphi \mapsto \varphi(x)$ is thus a linear and bounded functional on $\widetilde{\mathcal{H}}^K(\Omega)$ (for any $x \in \Omega$).

We have defined a pre-Hilbert space, that is, a vector space endowed with a scalar product and, in order to obtain a complete metric space, the completion of the pre-Hilbert space $\widetilde{\mathcal{H}}^K(\Omega)$ is considered by taking all linear combinations based on countably many points $Y = (y^1, y^2, \dots)$ in Ω. The corresponding space is denoted by $\mathcal{H}^K(\Omega)$ and is the *reproducing Hilbert space* generated from the kernel K.

Clearly, both properties (2.10) remain true in $\mathcal{H}^K(\Omega)$, and we also observe that

$$\begin{aligned} |\varphi(x) - \varphi(y)| &= |\langle K(\cdot, x) - K(\cdot, y), \varphi \rangle_{\mathcal{H}^K(\Omega)}| \\ &\lesssim \|K(\cdot, x) - K(\cdot, y)\|_{\mathcal{H}^K(\Omega)} \|\varphi\|_{\mathcal{H}^K(\Omega)} = D(x, y) \|\varphi\|_{\mathcal{H}^K(\Omega)}, \qquad x, y \in \Omega. \end{aligned} \tag{2.11}$$

Since K and thus D are continuous in Ω, all functions in $\mathcal{H}^K(\Omega)$ are continuous, at least.

Discrepancy functional. Consider the integration error in (2.2), which is split into two parts, namely

- a contribution estimating the *regularity of the function* φ under consideration, which is is independent of the choice of the interpolation points, and
- a contribution depending solely upon the kernel K and the mesh points, which is independent of the choice of the function.

Various equivalent formulations of the second term can be derived [17], in physical variables, in spectral variables associated with the kernel, as well as in discrete Fourier variables associated with a lattice. Although these formulations are in principle equivalent, they shed a very different light on the problem. For the factorization in physical variables, we find (2.3) as stated earlier.

2.3 Definition of discrete projection and differential operators

A scale of finite dimensional spaces. Let us summarize our notation in the finite dimensional setup, in which we now fix an integer N, a finite collection of points $Y = (y^1, \dots, y^N)$ in Ω, together with an admissible K on Ω. Based on these data, we define the finite dimensional vector space $\mathcal{H}_Y^K(\Omega)$ consisting of all linear combinations of the so-called basis functions $x \mapsto K(x, y^n)$, that is,

$$\mathcal{H}_Y^K(\Omega) = \Big\{ \sum_{1 \le m \le N} a_m K(\cdot, y^m) \,/\, a = (a_1, \dots, a_N) \in \mathbb{R}^N \Big\}. \tag{2.12}$$

Since the kernel K is continuous, $\mathcal{H}_Y^K(\Omega)$ embeds into the space of all continuous functions on Ω. We consider the bilinear expression (2.2) and the corresponding Hilbert space $\mathcal{H}_Y^K(\Omega)$ with norm $\|\varphi\|^2_{\mathcal{H}_Y^K(\Omega)} = a^T K(Y,Y) a$. The reproducing kernel property (immediate from (2.9))

$$\langle K(\cdot, y^m), K(\cdot, y^n) \rangle_{\mathcal{H}_Y^K(\Omega)} = K(y^m, y^n), \tag{2.13}$$

allows one to relate the coefficients a_m of the decomposition of a function $\varphi = \sum_{1 \le m \le N} a_m K(\cdot, y^m)$ to its scalar product with the basis functions, namely

$$\begin{aligned} \varphi(y^n) = \langle \varphi, K(\cdot, y^n) \rangle_{\mathcal{H}_Y^K(\Omega)} &= \sum_{1 \le m \le N} a_m \langle K(\cdot, y^m), K(\cdot, y^n) \rangle_{\mathcal{H}_Y^K(\Omega)} \\ &= \sum_{1 \le m \le N} a_m K(y^m, y^n) = a^T K(Y, y^n), \end{aligned} \tag{2.14}$$

which is nothing but the standard scalar product of the vectors a and $K(Y, y^n)$ (for any given n).

K-projection. Given any continuous function f on Ω), the vector $f_Y = \big(f(y^1), \dots, f(y^N)\big)$ consists of the values of this function at the given points. We define $\mathbf{P}_Y^K(f)$ in the space $\mathcal{H}_Y^K(\Omega)$ by reconstructing a suitable element of $\mathcal{H}_Y^K(\Omega)$ from the given vector f_Y, specifically we set

$$\mathbf{P}_Y^K(f)(x) = a^T K(x, Y) = \sum_{1 \le n \le N} a_n K(x, y^n), \qquad x \in \Omega, \qquad a = K(Y,Y)^{-1} f_Y. \tag{2.15a}$$

Clearly, $(\mathbf{P}_Y^K \circ \mathbf{P}_Y^K)(f) = \mathbf{P}_Y^K(f)$ and, moreover, $\mathbf{P}_Y^K(\varphi) = \varphi$ for any function $\varphi = a^T K(\cdot, Y) \in \mathcal{H}_Y^K(\Omega)$. The norm of the K-projection operator $\mathbf{P}_Y^K$ is

$$\|\mathbf{P}_Y^K(f)\|^2_{\mathcal{H}_Y^K(\Omega)} = f_Y^T K(Y,Y)^{-1} f_Y. \tag{2.15b}$$

A basis of functions is naturally associated with the discrete space $\mathcal{H}_Y^K(\Omega)$, and consists of the set of N functions $\theta_Y^n : \Omega \to \mathbb{R}$ taking the values 0 or 1 at the points of the set Y. Precisely, using the notation $\theta_Y = (\theta_Y^1, \dots, \theta_Y^N)$, we define the **$K$-basis** (as a whole) by

$$\theta_Y(x) = K(Y,Y)^{-1} K(Y, x), \qquad x \in \Omega. \tag{2.16}$$

It follows that, as expected,

$$\big(\theta_Y(y^m)\big)_{1\le n,m\le N} = K(Y,Y)^{-1}K(Y,Y) = \mathrm{Id} \tag{2.17a}$$

(the identity matrix) or $\theta_Y^n(y^m) = \delta_{nm}$ (using Kronecker notation). On the other hand, the scalar product of any two basis functions is

$$\big\langle \theta_Y^m, \theta_Y^n \big\rangle_{\mathcal{H}_Y^K(\Omega)} = K^{-1}(y^m, y^n), \tag{2.17b}$$

where $K^{-1}(y^m, y^n)$ stands for the (n,m)-coefficient of the matrix $K(Y,Y)^{-1}$. Importantly, this partition of unity is useful in expressing the discrete projection of a function $f : \Omega \to \mathbb{R}$, namely

$$\mathbf{P}_Y^K(f) = \sum_{1\le n\le N} f(y^n)\theta_Y^n. \tag{2.17c}$$

K-gradient operator. Once more, we fix a set of points $Y = (y^1, \dots, y^N)$ in Ω and we are interested in functions in the associated discrete space $\mathcal{H}_Y^K(\Omega)$. Given a continuous function $h : \Omega \to \mathbb{R}$, by using our notation $h_Y = h(Y) = (h(y^1), \dots, h(y^N)) \in \mathbb{R}^N$ we introduce the K-gradient $\nabla_Y^K h$ of the function h, defined by taking the gradient of the basis function as the D-vector-valued function

$$(\nabla_Y^K h)(x) = \sum_{n=1,\dots,N} h(y^n)\nabla\theta^n(x) = (\nabla K_Y)(x)K(Y,Y)^{-1}h_Y \in \mathbb{R}^D, \qquad x \in \Omega, \tag{2.18a}$$

where $(\nabla K_Y)(x) = \big(\nabla K(y^1, x), \dots, \nabla K(y^N, x)\big) \in \mathbb{R}^{N\times D}$ is a rectangular matrix. This definition can be cast in the form of an operator, namely with

$$\nabla K(Y,Y) = \big(\partial_d K(y^n, y^m)\big)_{d=1,\dots,D;n,m=1,\dots,N} \in \mathbb{R}^{D\times N\times N} \tag{2.18b}$$

the **K-gradient operator** ∇_Y^K reads

$$\nabla_Y^K = \big(\partial_d K(Y,Y)K(Y,Y)^{-1} \in \mathbb{R}^{N\times N}\big)_{d=1,\dots,D} = \nabla K_Y K_Y^{-1} \in \mathbb{R}^{D\times N\times N} \tag{2.18c}$$

Other K-operators. We can also define many other integro-differential operators of interest in the applications. For instance, we can introduce the **K-divergence operator** $(\nabla_Y^K)^T$, where the transposition is defined with respect to the standard Euclidian product, i.e. for any vectors $h_Y \in \mathbb{R}^N$ and $f_Y \in \mathbb{R}^{N\times D}$ we write

$$< h_Y, (\nabla_Y)^T f_Y >_{\mathbb{R}^N} = < \nabla_Y^K h_Y, f_Y >_{\mathbb{R}^{N\times D}} . \tag{2.19}$$

The connection with a weak formulation for functions defined on $\mathbb{R}^D$ is as follows. For any measure μ and scalar-valued function $h : \mathbb{R}^D \mapsto \mathbb{R}$, and a smooth vector field f on $\mathbb{R}^D$, we have

$$< \nabla h, f\mu >_{\mathcal{D},\mathcal{D}'} = \int_{\mathbb{R}^D} < \nabla h, f >_{\mathbb{R}^D} d\mu = -\int_{\mathbb{R}^D} h\big(\nabla \cdot f d\mu\big). \tag{2.20}$$

Thus, $(\nabla_Y^K)^T f \simeq -\big(\nabla \cdot f d\mu\big)$ is meaningful in a weak sense with the choice $\mu = \frac{1}{N}\sum_{n=1}^N \delta_{y^n}$. Observe that this notion is not fully consistent with the standard divergence operator but depends upon the kernel.

The **K-Laplacian operator** defined as $\Delta_Y^K = \nabla_Y^K \cdot \nabla_Y^K \in \mathbb{R}^{N\times N}$ is the discrete counterpart of the standard Laplacian operator Δ, and here we use contraction with respect to the indices $d = 1, \dots, D$ and matrix multiplication in the indices $n = 1, \dots, N$.

The **K-Hessian operator** defined as $\nabla_Y^{K,2} = \nabla_Y^K \otimes \nabla_Y^K \in \mathbb{R}^{D\times D\times N\times N}$ is the discrete counterpart of the standard Hessian operator ∇^2 and involves a standard matrix multiplication in the two indices ranging $1, \dots, N$.

Finally, for any discrete vector field $S_Y \in \mathbb{R}^{N\times D}$ we introduce

$$\mathbf{\Pi}_Y^K S_Y = \big(\nabla_Y^K\big((\nabla_Y^K)^T\nabla_Y^K\big)^{-1}(\nabla_Y^K)^T S_Y\big), \qquad \zeta_Y = \mathcal{L}_Y S_Y = \big(\mathrm{Id} - \mathbf{\Pi}_Y^K\big)S_Y, \tag{2.21a}$$

and obtain the orthogonal **K-Hodge decomposition**

$$S_Y = \nabla_Y^K(h_Y + f_Y) + \zeta_Y, \qquad (\nabla_Y^K)^T\zeta_Y = 0, \qquad (\nabla_Y^K)^T\nabla_Y^K f_Y = 0, \tag{2.21b}$$

where f_Y representss a discrete harmonic component. This decomposition is modeled upon the Hodge decomposition

$$S = \nabla(h + h_0) + \zeta, \qquad \nabla \cdot \zeta\mu = 0, \qquad \zeta \cdot \eta = 0, \qquad \nabla \cdot (\nabla h\mu) = 0, \tag{2.21c}$$

where h_0 has compact support and $\mu \simeq \frac{1}{N}\sum_{n=1}^N \delta_{y^n}$.

2.4 Definition of discrete interpolation and extrapolation operators

Setup. An admissible kernel K is fixed throughout. Consider an arbitrary collection of M points Y in $\Omega \subset \mathbb{R}^D$ and a vector-valued function $f = f(y) \in \mathbb{R}^P$, which provide us with the two sets of data $Y \in \mathbb{R}^{M\times D}$ and $f_Y \in \mathbb{R}^{M\times P}$. Consider also a collection X of N points in $\Omega \subset \mathbb{R}^D$ with $N << M$. Let us associate with these two sets (as in (2.12)) the corresponding functional spaces $\mathcal{H}^K_Y$ and $\mathcal{H}^K_X$, which provides us with the framework in which our definitions now make sense. Finally we introduce the rectangular matrix consissting of all relevant values of the kernel

$$K(X,Y) = \big(K(x^n, y^m)\big)_{\substack{1\leq n\leq N\\ 1\leq m\leq M}} \in \mathbb{R}^{N\times M}, \tag{2.22a}$$

together with its discrete gradient

$$\nabla K(X,Y) = \big(\nabla_{y^m} K(x^n, y^m)\big)_{\substack{1\leq n\leq N\\ 1\leq m\leq M}} \in \mathbb{R}^{N\times D\times M}. \tag{2.22b}$$

K-extrapolation. Suppose that we are only given some data $f_X \in \mathbb{R}^{N\times T}$ on a small set of points X. Then we can extrapolate these values on a larger set of points Y by introducing

$$\mathbf{Ext}^K_{X,Y} f_X = K(Y,X)K(X,X)^{-1} f_X \in \mathbb{R}^{M\times T}. \tag{2.23}$$

Similarly, the K-extrapolation of the gradient operator on Y is defined as

$$\nabla\mathbf{Ext}^K_{X,Y} f_X = (\nabla K)(Y,X)K(X,X)^{-1} f_X \in \mathbb{R}^{M\times D\times T}. \tag{2.24}$$

K-interpolation. Conversely, suppose that we are given some data $f_X \in \mathbb{R}^{N\times P}$ on a set $X \in \mathbb{R}^{N\times D}$. Then for any collection $Y \in \mathbb{R}^{M\times D}$, we define $\mathbf{Int}^K_{X,Y} f_X \in \mathbb{R}^{M\times P}$ by the minimization problem

$$\mathbf{Int}^K_{X,Y} f_X = \arg \inf_{g_Y \in \mathbb{R}^{M\times P}} \|K(Y,X)K(Y,Y)^{-1} g_Y - f_X\|, \tag{2.25a}$$

which yields us

$$f_Y = \mathbf{Int}^K_{X,Y} f_X = K(Y,Y)K(X,Y)^{-1} f_X \in \mathbb{R}^{M\times P}. \tag{2.25b}$$

Here, by definition, $K(X,Y)^{-1}$ is the standard (least-square) inverse matrix, characterized by the conditions

$$\begin{aligned} K(X,Y)^{-1} &= \big(K(Y,X)K(X,Y)\big)^{-1} K(Y,X) \in \mathbb{R}^{M\times N}, \qquad M \leq N,\\ K(X,Y)^{-1} &= K(Y,X)\big(K(X,Y)K(Y,X)\big)^{-1} \in \mathbb{R}^{M\times N}, \qquad N \leq M. \end{aligned}$$

In particular, we ensure the relation $\mathbf{Ext}^K_{Y,X} f_Y = f_X$ exactly.

K-projections. Interpolation and extrapolation are particular cases of the projection operators that we define now. Let $(X,Y,Z) \in (\mathbb{R}^{M\times D}, \mathbb{R}^{N\times D}, \mathbb{R}^{P\times D})$ be collections of D-vectors, and let us assume the restriction $M \leq N$. Then we define $\mathbf{\Pi}_{X,Y,Z}$ by

$$\mathbf{\Pi}^K_{X,Y,Z} = \mathbf{Ext}^K_{Y,Z}\mathbf{Int}^K_{X,Y} f_Y = K(Z,Y)K(X,Y)^{-1}. \tag{2.26}$$

For instance, $\mathbf{Ext}^K_{X,Y} = \mathbf{\Pi}^K_{X,X,Y}$ and $\mathbf{Int}^K_{X,Y} = \mathbf{\Pi}^K_{X,Y,Y}$. A particularly interesting operator is $\mathbf{\Pi}^K_{X,Y,X}$, whose kernel "contains the information" lost from approximating a function from a larger set X to a smaller set Y.

2.5 A steepest descent algorithm

K-discrepancy error. Consider two sets of points X, Y as above. In agreement with (2.3), we compute the discrepancy error between the discrete measures μ_Y and μ_X as

$$\mathbf{d}^K(\mu_Y,\mu_X)^2 = \frac{1}{M^2}\sum_{l,m=1}^{M} K(y^l,y^m) + \frac{1}{N^2}\sum_{n,q=1}^{N} K(x^n,x^q) - \frac{2}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M} K(x^n,y^m). \tag{2.27}$$

The notion of discrepancy error $\mathbf{d}^K(\mu_Y,\mu_X)^2$ also allows to define *transportation maps,* which move a set of points $t\mapsto X_t$, with $X_0=X$, by

$$X_\infty = \arg_{X\in\mathbb{R}^{N\times D}} \inf \mathbf{d}^K(\mu_Y,\mu_X)^2 \tag{2.28}$$

We can explicitly compute the gradient of $\mathbf{d}^K(\mu_Y,\mu_X)^2$ with respect to X as

$$\nabla_{x^n}\mathbf{d}^K(\mu_Y,\mu_X)^2 = \frac{2}{N^2}\sum_{o=1}^N \nabla_{x^n}K(x^n,x^o) - \frac{2}{NM}\sum_{m=1}^M \nabla_{x^n}K(x^n,y^m). \tag{2.29}$$

Consequently, the following semi-discrete scheme corresponds to a steepest descent algorithm:

$$\frac{d}{dt}x_t^n = \frac{2}{NM}\sum_{m=1}^M \nabla_{x_t^n}K(x_t^n,y^m) - \frac{2}{N^2}\sum_{o=1}^N \nabla_{x_t^n}K(x_t^n,x_t^o), \tag{2.30}$$

with the initial data $X_0=X$. Summarizing in our notation, the scheme can be written as

$$\frac{d}{dt}X_t = \frac{2}{NM}\nabla_{X_t}K(X_t,Y)_3 - \frac{1}{N^2}\nabla_{X_t}K(X_t,X_t)_3, \tag{2.31}$$

where the notation $_3$ denotes the contraction in the third index. The trajectories $t\mapsto X_t\in\mathbb{R}^{N\times D}$ follow characteristics corresponding to optimal transportation maps.

Fitting functions. Given some data $Y\in\mathbb{R}^{M\times D}$ and $f_Y\in\mathbb{R}^{M\times P}$, let us consider $\mathbf{\Pi}^K_{Y,X,Y}$, denoted for short $\mathbf{\Pi}^K_{X,Y}$, to be the projection operator (2.26). Starting from $X_0=X\in\mathbb{R}^{N\times D}$, fitting a function corresponds to determine a distribution of points X_∞ satisfying

$$\|f_Y - \mathbf{\Pi}^K_{X_\infty,Y}f_Y\|^2 = \inf_{X\in\mathbb{R}^{N\times D}}\|f_Y-\mathbf{\Pi}^K_{X,Y}f_Y\|^2. \tag{2.32a}$$

We rewrite this minimization problem as a constrained problem, namely

$$\|f_Y - \mathbf{\Pi}^K_{X_\infty,Y}f_Y\|^2 = \inf_{X\in\mathbb{R}^{N\times D}}\|f_Y-\mathbf{Ext}^K_{Y,X)}f_X\|^2, \qquad f_X = I(X,Y)f_Y, \tag{2.32b}$$

where **Ext** denotes the K-extrapolation operator. During the minimization step, we consider that $f_X\in\mathbb{R}^{N\times P}$ is independent of X and we take the constraint into account. Observe that

$$\nabla_X\|f_Y-\mathbf{Ext}^K_{Y,X)}f_X\|^2 = \langle \mathbf{Ext}^K_{Y,X)}f_X - f_Y, \nabla_X\mathbf{Ext}^K_{Y,X)}f_X\rangle \in \mathbb{R}^{N\times D}, \tag{2.33}$$

where the right-hand side is a contraction between a $M\times P$ object and the discrete operator

$$(\nabla_X\mathbf{Ext}^K)_{Y,X}f_X \in \mathbb{R}^{M\times N\times D\times P}. \tag{2.34}$$

Steepest descent algorithm. A set of moving points $t\mapsto X_t$ being given, consider the error term

$$\mathbf{e}(X,Y) = \mathbf{Ext}^K_{Y,X}f_X - f_Y \in \mathbb{R}^{M\times P}. \tag{2.35a}$$

The following scheme corresponds to the steepest descent algorithm

$$\frac{d}{dt}X_t = -\langle \mathbf{e}(X_t,Y), (\nabla_{X_t}\mathbf{Ext}^K)_{Y,X_t)}f_{X_t}\rangle_{\mathbb{R}^{M\times P}}. \tag{2.35b}$$

Using (2.24), we compute

$$\begin{aligned}\nabla_X\mathbf{Ext}^K_{Y,X)} &= \nabla_X\big(K(Y,X)K(X,X)^{-1}\big) = (\nabla_X K)(Y,X)K(X,X)^{-1} + K(Y,X)\nabla_X(K(X,X)^{-1})\\ &= (\nabla_X K)(Y,X)K(X,X)^{-1} - K(Y,X)K(X,X)^{-1}(\nabla_X K)(X,X)K(X,X)^{-1}.\end{aligned} \tag{2.35c}$$

Using this computation, the scheme reads

$$\begin{aligned}\frac{d}{dt}X_t = &-\langle \mathbf{e}(Y,X_t), (\nabla_{X_t}K)(Y,X_t)K(X_t,X_t)^{-1}f(X_t)\rangle_{\mathbb{R}^{M\times P}}\\ &+\langle K(X_t,X_t)^{-1}K(X_t,Y)e(X_t,Y), (\nabla_{X_t}K)(X_t,X_t)K(X_t,X_t)^{-1}f(X_t)\rangle_{\mathbb{R}^{N\times P}}.\end{aligned} \tag{2.35d}$$

We observe that

$$K(X,Y)\mathbf{e}(X,Y) = (K(X,Y)K(Y,X))(K(X,Y)K(Y,X))^{-1}K(Y,X)f_Y - K(X,Y)f_Y = 0,$$

and we can write our scheme componentwise as

$$\frac{d}{dt}x_t^n = -\sum_{m,p=1}^{M,P} \mathbf{e}(X_t,Y)_{m,p}(\nabla_{x_t^n}K)(y^m,x_t^n)cf(X)_{n,p}. \tag{2.35e}$$

3 An algorithm for support vector machines in machine learning

3.1 Error estimates for support vector machine

Standpoint in this section. Our aim in this section is to formulate a support vector machine in the context of the kernel theory in Section 2, and emphasize the role of error estimates based on sharp discrepancy sequences. Interestingly, our formulation uses an intermediate space (of size denoted by N_y, below) which avoids the computation of the inverse of the kernel and therefore leads us to an algorithm that is linear with respect to, both, the input and the output variables. In the standard formulation, there is no such pivot space, while we observe here that this notion simplifies the computational complexity.

We thus consider neural networks based on the concept of support-vector machine (SVM). Our presentation in this section is based on the notion of discrete projections, interpolation, and extrapolation operators introduced in the previous section. Consequently, our algorithms come with quantitative error estimates of the type we investigated in [17].

Setup of interest. An admissible kernel $K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ being fixed once for all, we work with several sets of points, denoted by $X \in \mathbb{R}^{N_x\times D}$, $Y \in \mathbb{R}^{N_y\times D}$, and $Z \in \mathbb{R}^{N_z\times D}$. Using the kernel, we compute the $(N_x \times N_y)$ Gram matrix $K(X,Y) = \big(K(x^n,y^m)\big)_{\substack{1\le n\le N_x\\ 1\le n\le N_y}}$ associated with the sets X and Y. Assuming that $N_y \le N_x$, a **feed-forward SVM**, by definition, is the K-projection operator associated with the set of points X, Y, Z defined by the matrix

$$P(X,Y,Z) = K(Z,Y)\,K(X,Y)^{-1} \in \mathbb{R}^{N_z\times N_x}, \tag{3.1}$$

where the inverse of a (rectangular) matrix A is defined as usual by $A^{-1} = (A^TA)^{-1}A^T$.

To any vector-valued function $\varphi : \mathbb{R}^D \to \mathbb{R}^P$ and any set of D points X, we associate the matrix $\varphi(X) \in \mathbb{R}^{N_x\times P}$, and we introduce its K-projection or **prediction** as

$$\varphi_Z = P(X,Y,Z)\varphi(X) \in \mathbb{R}^{N_z\times P} \tag{3.2}$$

and we also have its discrete gradient

$$\nabla^K\varphi_Z = (\nabla_z K)(Z,Y)\,K(X,Y)^{-1}\varphi(X) \in \mathbb{R}^{D\times N_z\times P}. \tag{3.3}$$

As also explained in the previous section, from ∇^K, we can also compute other differential operators such as Δ^K and $\nabla^{K,2}$

Error estimate. Given a SVM denoted by $\mathcal{P}(x,y,z)$, the worst error estimate for the expectation of a function φ reads

$$\Big|\frac{1}{N_z}\sum_{n=1}^{N_z}\varphi(z^n) - \frac{1}{N_z}\sum_{n=1}^{N_z}\varphi_{z^n}\Big| \le \Big(\mathbf{d}^K\big(\mu_X,\mu_Y\big) + \mathbf{d}^K\big(\mu_Y,\mu_Z\big)\Big)\|\varphi\|_{\mathcal{H}^K(\mathbb{R}^d)}, \tag{3.4}$$

in which the discrepancy error is now given by the discrete distance

$$\mathbf{d}^K\big(\mu_X,\mu_Y\big)=\frac{1}{N_x^2}\sum_{n=1,m=1}^{N_x,N_x}K\big(x^n,x^m\big)+\frac{1}{N_y^2}\sum_{n=1,m=1}^{N_y,N_y}K\big(y^n,y^m\big)-\frac{2}{N_xN_y}\sum_{n=1,m=1}^{N_x,N_y}K\big(x^n,y^m\big). \tag{3.5}$$

The bound in (3.4) involves now *two* contributions, namely the relative distances between the three distributions μ_X,μ_Y,μ_Z, with the notation $\mu_X=\frac{1}{N}\sum_{=1}^{N_x}\mu_{x^n}$. On the other hand, the norm arising in (3.4) can be approximated by $\|\varphi\|^2_{\mathcal{H}^K}\simeq\varphi_Y^TK(Y,Y)^{-1}\varphi_Y$.

3.2 Kernel engineering

Adding kernels. Many operations are available on kernels, which allow one to generate many new kernels from more basic ones. The operations listed now preserve the positivity property required for an admissible kernel. Consider two kernels $k_i(x,y):\mathbb{R}^D\times\mathbb{R}^D\mapsto\mathbb{R}, i=1,2$, and according to (3.1) consider the two projection operators:

$$\mathcal{P}_i(x,y,z)=k_i(z,y)k_i(x,y)^{-1}\in\mathbb{R}^{N_z\times N_x}. \tag{3.6}$$

Then a natural operation, leading to k_1+k_2, from these two kernels consists in adding two kernels, as stated in terms of Gram matrix:

$$\mathcal{P}(x,y,z)=k(z,x)k(x,y)^{-1},\qquad k(x,y)=k_1(y,x)+k_2(y,x) \tag{3.7}$$

The space of functions generated by k_1+k_2 is thus $\big\{\sum_{1\leq m\leq N_x}a^m(k_1(\cdot,x^m)+k_2(\cdot,x^m))\big\}$.

Tensorial kernels. A second operation, denoted by $k_1\cdot k_2$ and defined from any two kernels, consists in multiplying these two kernels, as stated in terms of Gram matrix:

$$\mathcal{P}(x,y,z)=k(z,x)k(x,y)^{-1},\qquad k(x,y)=k_1(x,y)\cdot k_2(x,y), \tag{3.8}$$

the notation $\cdot$ standing for the Hadamard product of matrices. The space of functions generated by $k_1\cdot k_2$ is $\big\{\sum_{1\leq m\leq N_x}a^m(k_1(\cdot,x^m)k_2(\cdot,x^m))\big\}$.

Convolution kernels. This convolution k_1*k_2 is defined by multiplying the Gram matrix, as follows:

$$\mathcal{P}(x,y,z)=k(z,x)k(x,y)^{-1},\quad k(x,y)=k_1(x,y)k_2(y,y) \tag{3.9}$$

$k_1(x,y)k_2(y,y)$ holding for the standard matrix multiplication. When $k_1(x,y)=\varphi_1(x-y)$ and $k_2(x,y)=\varphi_2(x-y)$, then the space of functions generated by k_1*k_2 is $\big\{\sum_{1\leq m\leq N_x}a^m(k(\cdot,x^m))\big\}$, in which where $k(x,y)=\big(\varphi_1*\varphi_2\big)(x-y)$ is the standard convolution of the two kernels.

Pipe kernels. We define the "pipe kernel" $k_1|k_2$ by the projection operator (3.1) as follows :

$$\mathcal{P}(x,y,z)=\mathcal{P}_1(x,y,z)\pi_1(x,y)+\mathcal{P}_2(x,y,z)\big(\mathrm{Id}-\pi_1(x,y)\big), \tag{3.10}$$

where we introduce the projection operator $\pi_1(x,y)=k_1(x,y)k_1(x,y)^{-1}=\mathcal{P}_1(x,y,x)$. Hence, a pipe kernel is defined by *splitting the projection operator* into two parts; the first part being treated by one kernel, while the second kernel handling the remaining error. This decomposition is not a direct sum decomposition since the intersection of the two spaces may be non-trivial. Exchanging the role of the k_1 and k_2 the discretization is generally different. Typically we choose the integer N_y to be small in comparison with N_x and N_z. The corresponding space of functions is now generated by $\sum_{1\leq m\leq N_x}a^mk_1(\cdot,x^m)+\sum_{1\leq m\leq N_x}b^mk_2(\cdot,x^m)$, hence the number of coefficients is now doubled. We also introduce its inverse concatenating matrix

$$k^{-1}(x,y)=\big(k_1(x,y)^{-1},k_2(x,y)^{-1}\big(I_{N_x}-\pi_1(x,y)\big)\big)\in\mathbb{R}^{2N_y\times N_x},$$

and the Gram matrix associated with a pipe kernel is $k(x,y)=\big(k_1(x,y),k_2(x,y)\big)\in\mathbb{R}^{N_x\times 2N_y}$.

Piping scalar product kernels: an example with polynomial regression. Let $S : \mathbb{R}^D \mapsto \mathbb{R}^N$ be a family of N basis functions φ_n, that is, $S(x) = \big(\varphi_1(x), \dots, \varphi_N(x)\big)$ and let us introduce a new kernel, called the *dot product kernel* (which is only conditionally positive definite) by

$$k_1(x, y) =< S(x), S(y) > . \tag{3.11}$$

Then, given any positive kernel $k_2(x, y)$ we can consider the corresponding pipe kernel $k_1|k_2$. In particular, such a construction is useful when a polynomial basis $S(x) = (1, x_1, \dots)$ is chosen: this produces a classical polynomial regression, allowing us to perfectly match the moments of distributions, with the remaining error being handled by the second kernel.

Neural networks as kernel methods. Our setup also encompasses the strategy classical adopted in the theory of deep learning, which are based on the notion of neural network. Namely, let us consider any feed-forward neural network of depth M, defined by

$$z_m = y_m g_{m-1}(z_{m-1}) \in \mathbb{R}^{N_m}, \qquad y_m \in \mathbb{R}^{N_m \times N_{m-1}}, \qquad z_0 = y_0 \in \mathbb{R}^{N_0}, \tag{3.12}$$

in which $y_0, \dots, y_M$ are our *weights* and g_m represents *prescribed activation functions*. By concatenation, we arrive at the function

$$z_M(y) = y_M z_{M-1}(y_0, \dots, y_{M-1}) : \mathbb{R}^{N_0 \times \dots \times N_M} \mapsto \mathbb{R}^{N_M}, \tag{3.13}$$

and this neural network is thus entirely represented by the kernel composition

$$K(y_m, \dots, y_0) = K_m\Big(y_m, K_{m-1}\big(\dots, K_1(y_1, y_0)\big)\Big) \in \mathbb{R}^{N_m \times \dots \times N_0}, \tag{3.14}$$

where $k_m(x, y) = g_{m-1}(xy^T)$, with indeed $z_M(y) = y_M k(y_{M-1}, \dots, y_0)$, as is relevant in kernel engineering.

3.3 Numerical results

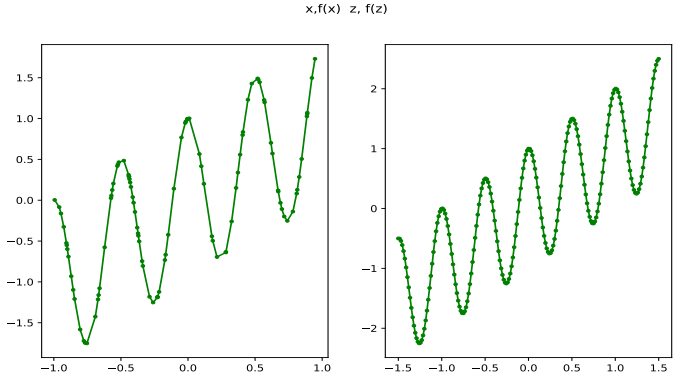
Dimension 1. We present a rather basic test, while referring to [18] for further numerical examples. We consider the role of kernels and we illustrate graphically the extrapolation operator (2.23) as well as the discrepancy error and functional norm computed from the kernels. We recall our notation: $x \in \mathbb{R}^{N_x \times D}$ and $f(x) : \mathbb{R}^{N_x \times D_f}$ for the training set and the function values, respectively. On the other hand, $z \in \mathbb{R}^{N_z \times D}$ and $f(z) : \mathbb{R}^{N_z \times D_f}$ represent the test set and the reference values respectively. Our test is based on the following data

$$\text{a function f}\ , \qquad \text{a kernel k}\ , \qquad \text{five integers } D, N_x, N_y, N_z, D_f,$$

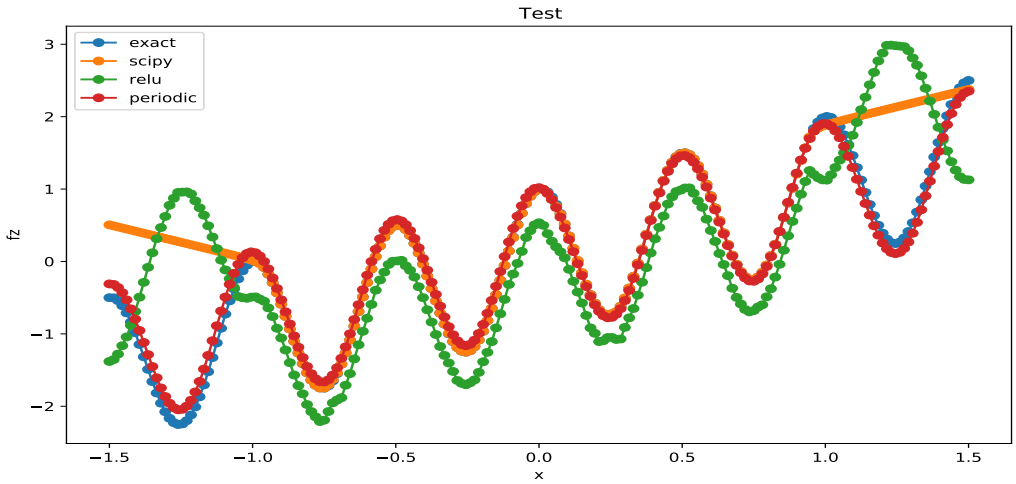
and, for definiteness, we consider the function

$$f(x) = \prod_{d=1,\dots,D} \cos(4\pi x_d) + \sum_{d=1,\dots,D} x_d. \tag{3.15}$$

We use a random generator, configured to select points $x \in [-1, 1]^{N_x \times D}$ randomly in the unit cube, and we pick up $z \in [-1.5, 1.5]^{N_x \times D}$ as uniformly distributed over another cube, to observe extrapolation effects. The values of x, z as well as corresponding values function are plot in the following figure:



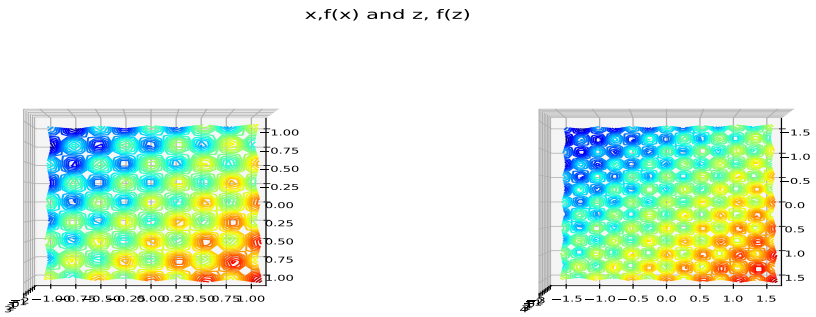
We then test the extrapolation operator (2.26) using three different kernels. The first one is a Gaussian kernel, named scipy, as we compared its results with the scipy implementation to check our results. The second one is a kernel equivalent to a RELU network. Finally the third one is a gaussian, periodic kernel. This kernel obviously gives more accurate results for this test.



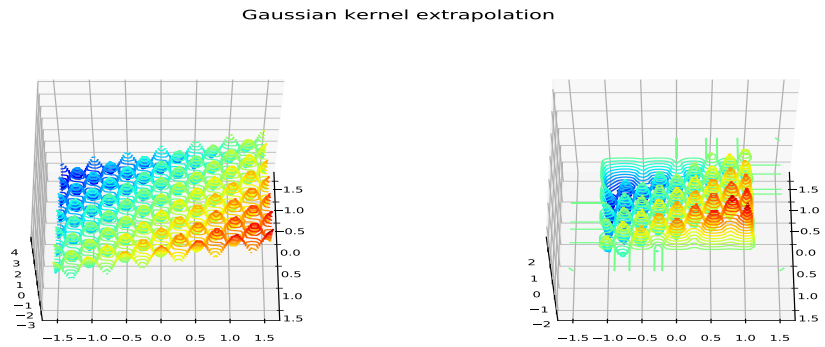
Finally, the following table shows the discrepancy error, as well as the kernel-norm of the computed functions:

	Scipy	RELU	periodic
disc. error	0.336247	0.2725986	0.1282466
function norm	0.7779849	0.7779849	0.7779849

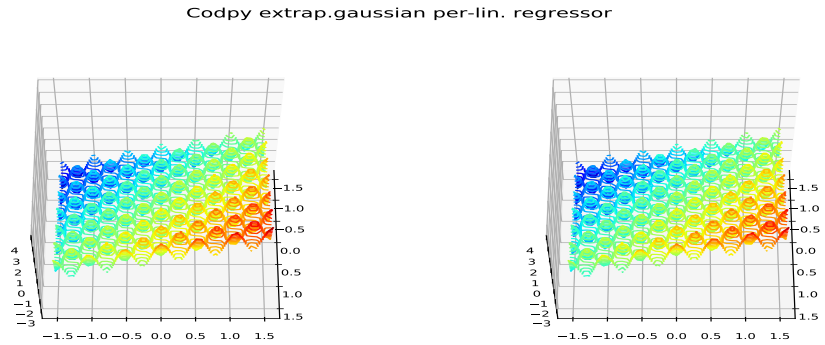
Dimension 2. The previous test can be formulate in any dimensions, and we show here the two-dimensional case ($D = 2$). The plot of $x, z, f(x), f(z)$ is as follows:



The following picture allows us to illustrate the effect of extrapolating data outside of the training set, by using a Gaussian kernel:



The following figure shows that the extrapolation is better dealt with a periodic kernel :



Finally, we provide the error analysis in the following table:

	Gaussian	periodic
disc. error	0.2105513	0.05348524
function norm	8181.435	2.000

3.4 Pattern recognition problem with MNIST

Algorithm applied to the MNIST database. We now use our algorithm for solving a very standard problem in machine learning. The MNIST database [13] which provides one with a real-word test for comparing numerical algorithms of pattern recognition. We build upon the discussion in Section 3.1 and consider a typical setup.

- We are given a *training set of* $M = 60000$ *images*, each representing $T = 10$ handwritten digits. To each image we attach a label which is an integer between 0 and 9. Each image is a matrix of 28×28 scalars representing gray scale, which we display as a D-vector in dimension $D = 784$.
- We also consider a *prediction set* consisting of $P = 10000$ images, which we will use to compute the error of our prediction.
- Moreover, we introduce a *regression set* consisting of N *randomly chosen images* (with $N \leq M$) among the M images of our training set.

We use the notation $N = N_Y$, $P = N_Z$, and $M = N_X$. and we summarize our notation in the following table:

$X \in \mathbb{R}^{M \times D}$	$Y \in \mathbb{R}^{N \times D}$	$Z \in \mathbb{R}^{P \times D}$	$L_X \in \mathbb{R}^M$	$L_Y \in \mathbb{R}^N$	$L_Z \in \mathbb{R}^P$
training images	regression	prediction	training labels	regression	prediction

Table 3.3: keras - scores and times

M/N - score	200	400	600
10000	0.9494	0.9503	0.9525
20000	0.9666	0.9689	0.9667
30000	0.9724	0.9755	0.9778
40000	0.9754	0.9767	0.9793
50000	0.9786	0.9807	0.9794
60000	0.9786	0.9815	0.9783

M/N - time	200	400	600
10000	2.26	3.13	3.63
20000	3.08	4.90	5.74
30000	4.95	8.54	9.22
40000	5.66	9.46	10.89
50000	8.03	13.40	14.54
60000	8.24	14.15	16.84

Throughout, an admissible kernel denoted by K is fixed. To any discrete vector-valued function $f_X \in \mathbb{R}^{M\times T}$, as explained earlier we associate the *extrapolation function*

$$f_X(Z) = K(Z,X)K(X,X)^{-1}f_X \in \mathbb{R}^T, \qquad Z \in \mathbb{R}^D. \tag{3.16}$$

Once the training of a given model L is performed, we predict its value, denoted $L(z)$ for an image z. The accuracy of the numerical output is evaluated by computing

$$E(N,M,L) = \frac{1}{P}\#_{z\in Z}\big\{z \,/\, L_Z(z) = L(z)\big\}. \tag{3.17}$$

An illustrative test. Table 3.3 show the results based on a neural network of N dense layers, trained with M images, with Adam optimizer and sparse categorical cross-entropy loss function[1]. We will use this test as a comparison for our results, since we can implement it on the same computer and is CPU-parallel, as are all of our tests. Our numerical tests are based on a quite standard kernel, that is, the transported Gaussian kernel

$$K(x,y) = \exp\Big(-C\,|\mathrm{erf}^{-1}(2x-1) - \mathrm{erf}^{-1}(2y-1)|^2\Big), \tag{3.18}$$

where erf^{-1} denote the inverse of the standard error function and C is a scale factor. Using transported kernels is natural in the present setting, since these kernels are compactly supported as are our data, after a rescaling to the set $[0,1]^{784}$. In fact, we present our results with the kernel giving the best results among all other kernels we have tested (matérn, tensorial matérn, multiquadrics, RELU, linear regression kernel). However, all numerical results are quite comparable, except for linear regression kernels which exhibit much lower accuracy.

Extrapolation. A first idea is to extrapolate a category function $f_X \in \{0,\dots,T\}$ on the training set Z according to (2.23). However, we observed a rather poor accuracy with this strategy, which is not consistent with re-labelling category invariance. Instead, let us consider the following partition of unity, where $T = 10$ is the number of categories referred to in the literature as *one-hot encoding*:

$$f(x) = \big(f_1(x),\dots,f_T(x)\big) \in \mathbb{R}^T, \quad f_t(x) = \begin{cases} 1 & \text{when } L(x) = t, \quad x \in X, \\ 0 & \text{otherwise.} \end{cases} \tag{3.19}$$

Here we denote by $z \mapsto L(z) \in \{0,\dots,9\}$ the function returning the classification for any image $z \in [0,1]^D$, and the property $f(z) \in \{0,1\}^T$ holds for the function corresponding to this partition.

Consider extrapolating this function on the set Z using (2.23), that is, in this context

$$f_X(Z) = \mathbf{Ext}^K_{X,Z} f_X(X) = K(Z,X)K(X,X)^{-1}f_X(X) \in \mathbb{R}^{P\times T}. \tag{3.20}$$

The prediction being then computed as $L_X(z) = \arg\max_t(f_{X,1}(z),\dots,f_{X,t}(z),\dots,f_{X,T}(z))$. For this test, only the parameter M is needed. The results are summarized in Table 3.4. Observe that this extrapolation algorithm is very

[1] as define at https://www.tensorflow.org/tutorials/quickstart/beginner

Table 3.4: extrapolation - scores and times

M	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
scores	0.93	0.9492	0.9551	0.9619	0.9649	0.9666	0.9692	0.9707	0.9708	0.9726
times	1.26	2.83	5.12	8.53	11.10	16.28	22.41	30.37	40.06	48.68
$1 - \mathbf{d}^K(X, Z)$	0.9194	0.9384	0.9534	0.9537	0.9610	0.9561	0.9580	0.9586	0.9598	0.9654

Table 3.5: Projection - scores and times

M/N - score	1000	2000	3000
5000	0.9619	0.9634	0.9632
10000	0.9654	0.9701	0.9695
15000	0.9686	0.9711	0.9739
20000	0.97	0.9735	0.9753
25000	0.9713	0.9742	0.9766
30000	0.9725	0.9753	0.978

M/N - time	1000	2000	3000
5000	2.65	5.38	8.62
10000	4	7.97	12.74
15000	5.34	10.89	17.49
20000	6.83	13.79	22.12
25000	8.28	16.77	27.06
30000	9.70	19.71	31.83

simple, and has cubic complexity of order $O(M^3) + O(MP^2)$. The table also displays the discrepancy error (2.32), and clearly in this case the discrepancy error is similar to the score error.

The interpretation is the following : as observed earlier, we have the rough estimate of the error in (3.17) (where C_1, C_2 are constants):

$$\begin{aligned} E(M, M, L) = \|1_{L=L_X}\|_{\ell^1(Z)} &\leq C_1 \|f - f_X\|_{\ell^1(Z,\ell^\infty(T))} \leq C_2 \|f - f_X\|_{\ell^1(Z)} \\ &\leq C_2 \, \mathbf{d}^K(\mu_X, \mu_Z) \|f\|_{\mathcal{H}_X^K}, \end{aligned} \tag{3.21}$$

where we use the notation ℓ^p to denote the standard (integral or pointwise) discrete norm. We can consider here of several optimization choices based on this expression. For instance, we can try to reduce $\|f\|_{\mathcal{H}_X^K}$ by picking up a more adapted kernel K, which an idea closely related to *variance reduction for Monte-Carlo sampling*. This kernel engineering approach includes *data filtering*, and does lead to accuracy improvements, however at a cost of tuning our learning machine to input data which may not be always recommended. We will not explore this direction here, since we want to perform comparisons with standard methods. A second possibility is to choose the sampling and training set by reducing the distance $\mathbf{d}^K(\mu_X, \mu_Z)$. In our example however, we prefer to consider that these are fixed data.

Projection. We now study the effect of picking up a smaller set $Y \subset X$ in order to *reduce the complexity* of the vector machine algorithm. Our projection algorithm is based on interpolating the test-function over a smaller set $Y \in \mathbb{R}^{N \times D} \subset X \in \mathbb{R}^{M \times D}$ which is randomly selected among X, then using Y for extrapolation on the set Z using (2.25b). To summarize, in this context the predictor is

$$f_X(Z) = \mathbf{\Pi}_{X,Y,Z}^K f_X(X) = K(Z, Y) K(Y, X)^{-1} f_X(X) \in \mathbb{R}^{P \times T}, \tag{3.22}$$

where the kernel $K(Z, Y) \in \mathbb{R}^{P \times N}$. Table 3.5 contains the scores and times for values of N, M chosen in order to keep computational times comparable to our benchmark test. The resulting algorithm is quite simple to implement and enjoys two main features:

- For a given computational time, we observe that this projection method achieves better performance in comparison to a crude extrapolation algorithm, however at the cost of tuning the parameters N and M.
- This algorithm reduces the algorithmic complexity of the output machine $L(z)$.

Table 3.6: Matching algorithm - scores and times

M/N - score	200	400	600
10000	0.9396	0.9478	0.9509
20000	0.9433	0.9529	0.9556
30000	0.9416	0.9527	0.9581

M/N - time	1000	2000	3000
10000	36	47	58
20000	117	143	160
30000	251	286	312

Table 3.7: Projection algorithm - scores and times

M/N - score	200	400	600
10000	0.911	0.928	0.9357
20000	0.9153	0.9356	0.9437
30000	0.9182	0.9379	0.9457

M/N - time	1000	2000	3000
10000	0.86	1.29	1.75
20000	1.38	1.98	2.62
30000	1.79	2.60	3.47

Matching algorithm. We now consider the following numerical strategy: we use the projection operator (3.22) as a predictor, but we use Y as

$$Y = \arg \inf_{Y \in \mathbb{R}^{N \times D}} \mathbf{d}^K(\mu_{Y,f_Y}, \mu_{X,f_X})^2,$$

where the discrepancy error $\mathbf{d}^K(\mu_{Y,f_Y}, \mu_{X,f_X})^2$ corresponds to the discrepancy error of the tensor kernel constructed as the product of the linear regression kernel K_f and the Gaussian kernel. From the algorithmic point of view, this approach corresponds to matching each category by using (2.30), that is,

$$Y^p = \arg \inf_{Y^p \in \mathbb{R}^{N \times D}} \mathbf{d}^K(\mu_{Y^p,f_Y^p}, \mu_{X^p,f_X^p})^2, \quad p = 1 \dots 10,$$

which significantly reduces the overall algorithmic complexity. Table 3.6 displays our results with input data comparable to our benchmark test. In Table 3.7, we show a comparison with the projection algorithm for comparable inputs.

4 An algorithm for the Fokker-Plank-Kolmogorov system in mathematical finance

4.1 The fundamental equations

Stochastic differential equation. Let us begin with a description of the fundamental equations of mathematical finance, that is, the Fokker-Planck and Kolmogorov equations. We are interested in Markov-type stochastic processes, denoted by $t \mapsto X_t \in \mathbb{R}^D$ and governed by a stochastic differential equation (SDE). Specifically, we want to solve the *forward* initial value problem

$$\begin{aligned} dX_t &= r(t, X_t)dt + \sigma(t, X_t)dB_t, \\ X_t|_{t=0} &= X_0. \end{aligned} \tag{4.1}$$

Here, the initial data $X_0 \in \mathbb{R}^D$ is a prescribed random variable and $B_t \in \mathbb{R}^D$ denotes a (D-dimensional, with independent components) Brownian motion. We are also given a vector field $r \in \mathbb{R}^D$ representing the drift (or risk-free rate), together with a matrix-valued field $\sigma = \sigma(t, x)$ representing the random diffusion (or volatility). In our presentation we consider σ and r as given, while in specific applications it might be required to perform a calibration first and solve an inverse problem based on market data. A description of such a calibration algorithm can be found in [15].

Fokker-Planck equation. We are going to work with the description based on the partial differential equation associated with the SDE (4.1). We introduce the linear operator

$$\mathcal{L}\mu = \nabla \cdot (r\mu) + \nabla^2 \cdot (A\mu), \qquad A = \frac{1}{2}\sigma\sigma^T. \tag{4.2}$$

and consider the corresponding convection-diffusion equation. We recall that ∇ denotes the gradient operator, $\nabla\cdot$ the divergence operator, and $\nabla^2 = (\partial_i\partial_i)_{1\leq i,j\leq D}$ the Hessian operator. Given any parameter $(s,y)\in[0,+\infty)\times\mathbb{R}^D$, we denote by $(t,x)\mapsto\mu(t,x;s,y)$ (defined for $t\geq s$ and $x\in\mathbb{R}^D$) the *probability density measure* of the random variable X_t, *knowing* the value $X_s = y$ at the time s. By definition, this measure is the fundamental solution, defined from the base point (s,y), of the *Fokker-Planck problem*

$$\partial_t\mu - \mathcal{L}\mu = 0, \quad t\in[s,+\infty), \qquad \mu|_{t=s} = \delta_{x=y}. \tag{4.3}$$

The initial data in (4.3) is thus posed at the time $t = s$ and coincides with the Dirac measure $\delta_{x=y}$ at the point $x = y$.

By construction of the Fokker-Planck equation, the expectation of the random variable with respect to an arbitrary test-function φ is given as the corresponding moment of the measure μ. That is, the relation

$$\int_{\mathbb{R}^D}\varphi(t,x)\,d\mu(t,x;0,X_0) = \mathbf{E}^{X_t}(\varphi(t,\cdot)|X_0) \tag{4.4}$$

holds between the measure μ and the initial data X_0. We also observe that the solution to (4.3) is more naturally understood in the sense of distributions when the initial data is a measure. The solution μ is then a probability measure since the total mass is conserved in time, i.e.

$$\int_{\mathbb{R}^D}\mu(t,x;s,y)dx = 1. \tag{4.5}$$

Finally, we recall that the equation (4.3) also models the Brownian motion of particles in plasma physics.

It will be convenient to distinguish between two cases of interest in operational problems :

- The **martingale Fokker-Planck equation,** corresponding to the purely diffusive Fokker-Planck equation without transport part, that is, with $r\equiv 0$ in (4.1).
- The **general Fokker-Planck equation,** which includes both hyperbolic and parabolic parts, in which σ and r are prescribed functions.

Kolmogorov equation. The backward dual problem defined now provides us with the natural standpoint in mathematical finance. Taking the dual of the Fokker-Planck equation leads us to the *Kolmogorov equation* which is the central equation of interest, governing for instance for the price of option values, and is referred to as the *Black and Scholes equation*. It concerns an M-vector-valued unknown denoted by $P\mapsto P(t,x) = P(t,x;T,P_T)$, defined for all times $t\leq T$ and $x\in\mathbb{R}^D$, and is formulated as the following *backward problem*. Given a final time T and any (continuous and μ-integrable) final data P_T, we seek the solution to

$$\partial_t P - \mathcal{L}^* P = 0, \quad t\in[0,T], \qquad P|_{t=T} = P_T, \tag{4.6}$$

where $\mathcal{L}^*$ denotes the dual of the Fokker-Planck operator, given by

$$\mathcal{L}^* P = -r\cdot\nabla P + A\cdot\nabla^2 P. \tag{4.7}$$

In mathematical finance, the vector-valued function $P\in\mathbb{R}^M$ represents a portfolio of M instruments and M is typically a large integer. We recall the following terminology.

- The final data P_T is called the *payoff* of the instruments, whose underlyings are described by the random variable X_t satisfying the stochastic differential equation (4.1).
- The Kolmogorov equation (4.6) determines the so-called *forward values* (or price) $P(t,x)$ of the porfolio at any later time $t\in[0,T]$.
- In addition, the value at the time $t = 0$ is called the *fair value* $P_0 = P|_{t=0}$ of the portfolio.

Solving the Kolmogorov equation for a given portfolio of instruments allows us to determine not only its price, but also the whole of the *fair value surface* $(t,x)\mapsto P(t,x)$ (for all $t\in[0,T]$ and $x\in\mathbb{R}^D$). This function is important in many applications since standard risk measures are often determined from this surface; for instance, this is the case of risk measures of internal or regulatory nature, as well as of optimal investment strategies.

Martingale case. Let us consider here the martingale case and, for clarity in the presentation, let us repeat here the main equations. This case consists in solving the Fokker-Planck equation (4.2) and the Kolmogorov equation (4.6) without the advection part, that is,

$$dX_t = \sigma(t, X_t) dB_t, \qquad t \geq s \geq 0, \tag{4.8}$$

with initial conditions $X_0 \in \mathbb{R}^D$. The Fokker-Planck equation (4.2) for the probability measure $\mu(t, \cdot)$ reads (for $t \geq 0$)

$$\partial_t \mu = \nabla^2 \cdot (A\mu), \qquad A = \frac{1}{2}\sigma\sigma^T, \tag{4.9}$$

with initial conditions $\mu(0, \cdot) = \delta_{X(0)}$. In this martingale context, we then solve the Kolmogorov equation (4.6), that is, for some continuous and μ-integrable function P

$$\partial_t P - A \cdot \nabla^2 P = 0, \quad t \in [0, T], \qquad\qquad P|_{t=T} = P_T. \tag{4.10}$$

We will use the following definition associated with any martingale process $t \mapsto X_t \in \mathbb{R}^D$ (for $t \geq 0$). In the range $0 \leq s \leq t$ and $x, y \in \mathbb{R}^D$, we define the *transition probability operator* $\Pi = \Pi(s, t, x, y)$ as the solution to the Kolmogorov equation in the variables (t, y)

$$\partial_t \Pi = A \cdot \nabla_y^2 \Pi, \qquad \Pi(s, t, x, \cdot) = \delta_x, \qquad t \geq s \geq 0, \quad x \in \mathbb{R}^D. \tag{4.11}$$

It is not difficult to check the following property. If $t \mapsto X_t \in \mathbb{R}^D$ (for $t \geq 0$) is a martingale process and $\mu(t, \cdot)$ denotes its density measure, using transition operator Π we have

$$P(s, x) = \Pi(s, t, x, \cdot) * P(t, \cdot) = \mathbb{E}^{X_t}\Big(P(t, \cdot)|X_s = x\Big), \tag{4.12}$$

where $*$ denotes the convolution of two functions. The first identity holds since, by definition, the operator $\Pi(t, s, x, y)$ is the fundamental generator of the Kolmogorov equation, while the second identity is a consequence of Feynman-Kac's theorem.

4.2 The transport mesh-free method for mathematical finance

We now present our algorithm and solve numerically the equations (4.3) and (4.6), as follows. The method was first outlined in [15] and depends upon the a priori choice of an admissible kernel K of the type described in Section 2. We are going to compute a transport map, denoted below by $t \mapsto S(t, \cdot)$, which is naturally associated with the probability density μ of the stochastic process X_t satisfying (4.1) and associated with a random initial data X_0.

Step 1: Forward computation. Our first task is to determine the (almost) sharp discrepancy sequence that is most relevant for the simulation of the SDE problem (4.1). We introduce a semi-discrete scheme for the the Fokker-Planck equation (4.3), in which the time variable is kept continuous. This scheme provides us with an approximation of the solution μ as a sum of Dirac measures, namely

$$\mu_S(t) = \frac{1}{N}\sum_{n=1}^{N} \delta_{S^n(t)}, \qquad S(t) = (S^1, \ldots, S^N)(t), \tag{4.13}$$

in which $t \mapsto \big(S^n(t)\big)_{n=1,\ldots,N}$ in $\mathbb{R}^D$ is a suitable family of N "moving particles".

Specifically, a (small) approximation parameter $\epsilon \in (0, 1)$ being fixed, we solve the coupled system of ordinary differential equations (for $t \geq t_0 > 0$)

$$\begin{aligned} \frac{d}{dt} S &= r_S + \big(\nabla_S\big)^T A_S \in \mathbb{R}^{N \times D}, \\ S\big|_{t=t_0} &= (X_0, X_0, \ldots, X_0) + \sqrt{t_0}\, A(0, X_0)\mathcal{N}(0, 1), \end{aligned} \tag{4.14}$$

in which the function r and A in (4.3) are evaluated pointwise along the trajectories of the moving particles, as follows:

$$r_S(t) = \big(r(t, S^n(t))\big), \qquad A_S(t) = \big(A(t, S^n(t))\big)_{n=1,\ldots,N}. \tag{4.15}$$

Recall that the discrete differentiation $\big(\nabla_S\big)^T$ and, therefore, $\big(\nabla_S\big)^T A_S$ were defined in Section 2.3. In (4.14), X_0 stands for a specific realization of the random initial data, and in order to prevent the possibility of colliding particles, we have added a (vector-valued, centered, and of unit variance) i.i.d. random variable $\mathcal{N}(0, 1)$.

Step 1. Stability. The dynamics of particles is determined by the coupled system (4.14), and we have defined an approximation of the solution $\mu(t) \simeq \mu_S(t)$ to the forward Fokker-Planck problem (4.3). In the next section, we will check that this scheme is consistent and stable and, in fact, in the long time limit the sequence $S(t)$ approaches a sharp discrepancy sequence for the kernel K.

The above property has an important consequence, that is, we can rely on the framework developed in Section 2. In order to evaluate the accuracy of our strategy, at each discrete time we can compute the discrepancy error (2.3) and obtain the explicit error estimate

$$\Big| \int_{\mathbb{R}^D} \varphi(x) d\mu(t) - \frac{1}{N} \sum_{1 \leq n \leq N} \varphi(S^n(t)) \Big| \leq \mathbf{d}^K(\mu(t), \mu_S(t)) \, \|\varphi\|_{\mathcal{H}^K(\mathbb{R}^D)}, \tag{4.16}$$

valid for any moment of the measure μ and any time t (φ being an arbitrary test-function). Based on our analysis of the sharp discrepancy error, we can compare $\mathbf{d}^K(\mu(t), \mu_S(t))$ with the minimum value denoted (earlier on by $\mathbf{d}^K(\mu, \delta_{\overline{Y}})$ in Section 2) and, in turn, *explicitly check* the accuracy of the numerical solution.

Step 2. Backward computation. At this stage, we have computed an approximation of the sharp discrepancy sequence which is relevant (for a given realization of the initial data), and we are in a position to solve the backward Kolmogorov problem (4.6), using $t \mapsto S^n(t)$ (with $n = 1, \dots, N$) as a *moving grid*. We determine an function $t \mapsto P_S(t) \in \mathbb{R}^{N \times M}$ which is expected to be an approximation of the pointwise values $\big(P(t, S^n(t))\big)_{1 \leq n \leq N}$ of the Kolmogorov solution. As we further explain below, it can be computed as follows, between any two arbitrary times $s < t$ (for instance $0 = s < t = T$),

$$P_S(s) = \mathbf{\Pi}_S^{(t,s)} P_S(t), \qquad \mathbf{\Pi}_S^{(t,s)} = \big(\pi_{n,m}^{(t,s)}\big)_{1 \leq n,m \leq N}. \tag{4.17}$$

Here, the matrix $\mathbf{\Pi}_S^{(t,s)} \in \mathbb{R}^{N \times N}$ is provided *explicitly* (see below) and is nothing but the discrete fundamental operator associated with the Kolmogorov equation. Hence, this step of our numerical algorithm reduces to *computing the inverse of a matrix*. Further details are given in a following section where we study the consistency and convergence of the method.

Step 2. Stability. Interestingly, in the present context we can regard the matrix $\mathbf{\Pi}_S^{(t,s)}$ as a *Markov-chain process:* $\pi_{n,m}^{(t,s)}$ is the probability that the stochastic process will jump from the sharp discrepancy state $S^n(t)$ (at the time t) to the sharp discrepancy state $S^m(s)$ (at the time s). Our numerical algorithm provides this structure in a natural way, and $\pi_{n,m}^{(t,s)}$ turns out to be a bi-stochastic matrix (having each rows and each column summing to 1). This step of the algorithm is a martingale process which takes into account the diffusive part A of the SDE problem (4.1) but with the drift part r *suppressed* (since it was already handled in the first step). Importantly, the numerical solution enjoys the error estimate associated with the kernel K, that is,

$$\Big| \int_{\mathbb{R}^D} P(t, \cdot) d\mu(t, \cdot) - \frac{1}{N} \sum_{1 \leq n \leq N} P(t, S^n(t)) \Big| \leq \mathbf{d}^K(\mu(t), \mu_S(t)) \, \|P(t, \cdot)\|_{\mathcal{H}^K(\mathbb{R}^d)}, \tag{4.18}$$

which allows us to quantitatively determine, once more, the *accuracy of the numerical solution.*

We point out that we can also deal with partial derivative operators and, for instance, our framework allows us to compute forward sensitivities, that is,

$$\nabla P(s) \text{ is approximated by } \nabla_y P(t, y^n(t))_{1 \leq n \leq N}. \tag{4.19}$$

This allows us to determine, for instance, hedging strategies [26]. Motivated by fluid dynamics, we could also treat more complex operators such as the Hessian operator or the Helmholtz-Hodge decomposition.

Numerical illustration: the SABR model. We can illustrate a feature of our algorithm by considering the example of the (shifted) SABR model (see [1] and the references therein), described by the following two stochastic differential equations with unknown variables $X = (F, \alpha)$:

$$\begin{pmatrix} dF_t \\ d\alpha_t \end{pmatrix} = \rho \begin{pmatrix} \alpha_t (F_t + s)^\beta & 0 \\ 0 & \nu \alpha_t \end{pmatrix} \begin{pmatrix} dW_t^1 \\ dW_t^2 \end{pmatrix}, \tag{4.20}$$

supplemented with prescribed random initial conditions $X_0 = (F_0, \alpha_0)$ at the time $t = 0$. Here, $\beta \in [0, 1]$ is a parameter representing the so-called *constant elasticity of variance* (CEV) and the constant $\nu \geq 0$ is the volatility parameter, while W_t^1, W_t^2 are independent Brownian motions, while the so-called correlation matrix ρ is also prescribed.

Let us consider the transported Matérn kernel, described for instance in [17]. In Figure 4.1, we plot our approximation of the sharp discrepancy sequence associated with the SABR model. We use $N = 200$ points and the parameters $F_0 = 0.03$, $\alpha_0 = 0.1$, $\nu = 0.1$, $\beta = 1$, and $\rho_{12} = \rho_{21} = 0.5$ (with $\rho_{11} = \rho_{22} = 1$ for the correlation matrix). We plot the set of points $\big(y^1(t), \dots, y^N(t)\big)$, where the y-axis represents the volatility process α_t, and x-axis denotes the interest rates F_t. We observe that the initial support (an approximation of a Dirac measure) spread throughout the computational domain and expands as the time increases, according to the transport-diffusion process described by (4.20).

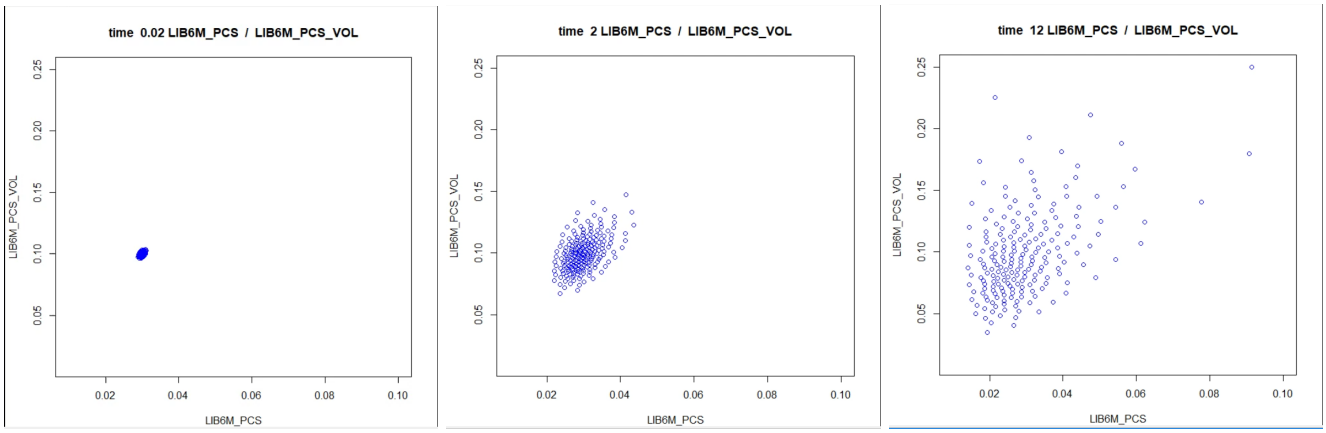


Figure 4.1: SABR model at the times $t = 0.02, 2, 12$ with $N = 200$ points.

4.3 Analysis of the semi-discrete scheme for the Fokker-Plank equation

Main statement for this section. The semi-discrete scheme (4.14) enjoys many properties (global existence, stability, convergence) which we now state and establish. We work with the following *modified Fokker-Planck problem*

$$\begin{aligned} &\partial_t \mu^{\mathrm{ex}} - \mathcal{L}\mu^{\mathrm{ex}} = 0, && t \in [t_0, +\infty),\ x \in \mathbb{R}^D, \\ &\mu^{\mathrm{ex}}|_{t=t_0} = \mu_{S_0^\star}, && S_0^\star = (X_0, X_0, \dots, X_0) + \sqrt{t_0}\, A(0, X_0)\mathcal{N}(0, 1), \end{aligned} \tag{4.21}$$

whose (exact) unknown is denoted by $\mu^{\mathrm{ex}} = \mu^{\mathrm{ex}}(t, x)$ and for which the initial data is posed at some time $t = t_0$ and randomly perturbed by a Brownian motion (as explained earlier). We impose a uniform ellipticity condition in (4.22) below. Thanks to the uniform parabolic property we can restrict attention to sufficiently regular solutions.

An important insight for our theory is provided by the transported version of the equation, stated in (4.30) in the proof below. Our main result concerning the first step of the algorithm is as follows.

Proposition 4.1 (Semi-discrete algorithm for the Fokker-Planck equation)**.** *Consider the Fokker-Plank problem* (4.21) *under the following ellipticity and linear growth conditions (with $\lambda, \alpha_-, \alpha_+ > 0$):*

$$|r(t, x) \cdot x| \leq \lambda |x|^2, \qquad \alpha_- |x|^2 \leq x^T \cdot A(t, x) \cdot x \leq \alpha_+ |x|^2, \qquad t \geq 0,\ x \in \mathbb{R}^D. \tag{4.22}$$

Then the semi-discrete scheme (4.14) *defines a sequence of moving particles $S(t) = (S^n(t))$ together with an associated measure $\mu_S(t) = \frac{1}{N} \sum_n \delta_{S^n(t)}$ which determines a convergent approximation of the solution $\mu^{ex}(t)$ to* (4.21)*, as follows.*

- **Sup-norm estimate.** *The map $t \mapsto S(t)$ is defined for all times and remains globally bounded in the sup-norm, i.e.*

$$\sup_{t \geq 0} e^{-\lambda t} \sum_{n=1}^N |S^n(t)|^2 \leq \sum_{n=1}^N |S^n(0)|^2, \tag{4.23}$$

(which is an analogue of the second moment $\int |x|^2 d\mu^{ex}$ for solutions to the Fokker-Plank equation).

- **Balance law of momentum.** *The sum of all components satisfies the evolution equation*

$$\frac{d}{dt}\sum_{n=1}^{N} S^n(t) = \sum_{n=1}^{N} r(t, S^n(t)), \qquad t \geq 0, \tag{4.24}$$

and, in particular, the total momentum is constant in time when the drift vanishes identically.

- **Self-adjointness property.** *The scheme* (4.14) *can be cast in the symmetric form*

$$\frac{d}{dt}S = r_S + B_S S, \tag{4.25}$$

in which $B_S = B_S(t)$ is a positive-definite symmetric matrix.

- **Error estimate.** *For any relevant test-function φ and any time $t \geq 0$ one has*

$$\left| \int_{\mathbb{R}^D} \varphi(x) d\mu^{ex}(t,x) - \frac{1}{N}\sum_{n=1}^{N} \varphi(S^n(t)) \right| \leq \mathbf{d}^K(\mu^{ex}(t), \mu_S(t))\, \|\varphi\|_{\mathcal{H}^K(\mathbb{R}^D)}, \tag{4.26}$$

where the discrepancy error was defined in Section 2.

Proof of the properties for the Fokker-Planck problem. We rely on a formulation of the Fokker-Plank equation with test-functions ϕ. The exact measure solution $t \mapsto \mu^{\mathrm{ex}}(t)$ satisfies

$$\frac{d}{dt}\int_{\mathbb{R}^D} \phi\, d\mu^{\mathrm{ex}}(t,\cdot) = \int_{\mathbb{R}^D} \Big(- r\,\nabla\phi + A\nabla^2\phi \Big)\, d\mu^{\mathrm{ex}}(t,\cdot). \tag{4.27}$$

We introduce the transport map $S^\star$ associated with the exact solution and defined by transporting the given initial data $\nu = \mu^{\mathrm{ex}}|_{t=t_0}$ by

$$S^\star_\#(t)\nu = \mu^{\mathrm{ex}}(t,\cdot) \tag{4.28}$$

and we rewrite the left-hand side of (4.27) in the form

$$\frac{d}{dt}\int_{\mathbb{R}^D} \phi\, d\mu^{\mathrm{ex}}(t,\cdot) = \frac{d}{dt}\int_{\mathbb{R}^D} \phi\circ S^\star d\nu = \int_{\mathbb{R}^D} \big(\nabla\phi\big)\circ S^\star\big)\cdot \partial_t S^\star\, d\nu.$$

On the other hand, the right-hand side of (4.27) reads

$$\begin{aligned} &\int_{\mathbb{R}^D} \Big(- r\,\nabla\phi + A\nabla^2\phi \Big)\, d\mu^{\mathrm{ex}}(t,\cdot) \\ &= -\int_{\mathbb{R}^D} ((\nabla\phi)\circ S^\star)\cdot (r\circ S^\star)\, d\nu - \int_{\mathbb{R}^D} \big((\nabla\phi)\circ S^\star\big)\cdot \big(\nabla\cdot\big((\nabla S^\star)^{-1}(A\circ S^\star)\big) d\nu. \end{aligned} \tag{4.29}$$

Here, we used integration by parts together with the identity $(\nabla\phi)\circ S^\star = (\nabla S^\star)^{-T}\nabla(\phi\circ S^\star)$.

This calculation allows us to propose a new formulation of the Fokker-Planck equation in terms of the transported map $S^\star$, namely:

$$\partial_t S^\star(t,\cdot) = -r(t,S^\star) - \nabla\cdot\big((\nabla S^\star)^{-1}(A\circ S^\star)\nu\big). \tag{4.30a}$$

By construction, μ^{ex} is a solution to (4.21) if and only if the the transport map $S^\star$ is a solution to (4.30a) and assumes the initial condition

$$S^\star(t_0, x) = x. \tag{4.30b}$$

Consequently, the discrete differential operator $\big(\nabla_S\big)^T A_S$ was defined to be consistent with $\nabla\cdot\big((A\circ S)\mu(t,\cdot)\big)$ (as is clear in view of (2.20)). Consequently, our scheme (4.14) is a discrete version of (4.30a) and therefore is formally consistent with the equation (4.30b).

Basic estimates. It is straightforward to multiply (4.14) by S and, by denoting ℓ^2 the standard Euclidian discrete norm, we obtain

$$\frac{d}{dt}\|S\|_{\ell^2}^2 =< S, \nabla_S^T A_S > + < S, r_S >=< \nabla_S S, A_S > + < S, r_S >\leq \lambda \|S\|_{\ell^2}^2,$$

in which $\nabla_S S = \mathrm{Id}$ is the discrete gradient operator applied to S and by construction is exact on polynomials of degree one. This also proves that the discrete scheme generates global-in-time solutions. The balance law of momentum is obtained by summation of all components of S. The self-adjointness property is established by observing that the scheme takes the form (4.25) in which

$$B_S = (\nabla_S)^T A_S (\nabla_S), \tag{4.31}$$

4.4 A semi-discrete scheme for the Kolmogorov equation

We present here some basic estimates enjoyed by our scheme and, in addition, numerical experiments demonstrate that it also enjoys better convergence rates in comparison to classical finite difference schemes. For simplicity in the presentation and without genuine loss of generality we assume that $M = 1$, that is, the main unknown P of the Kolmogorov equation is now scalar-valued. At this stage, we have determined a map $t \in [0,T] \mapsto S(t) \in \mathbb{R}^{N\times D}$ by the algorithm analyzed in Proposition 4.1 and we can thus turn our attention to solving the (transported version of the) Kolmogorov equation (4.6). In fact, since the transport part has been treated in the first step, we can focus our attention on the diffusion part, so without loss of generality we assume that the drift r vanishes identically and we study the martingale case (4.10). Hence, the Kolmogorov equation reads

$$\partial_t P - \mathcal{L}^* P = 0, \qquad \mathcal{L}^* P = A \cdot \nabla^2 P. \tag{4.32}$$

Our backward semi-discrete scheme for computing the solution $P = P(t,x)$ for $x \in \mathbb{R}^N$ and $t \in [0,T]$ reads

$$\frac{d}{dt}P = -B_S P, \qquad B_S = (\nabla_S^K)^T A_S (\nabla_S^K), \tag{4.33}$$

supplemented with the initial condition $P|_{t=T} = P_T(S(T))$ prescribed at the time T. Recall that the map $t \mapsto S(t)$ is computed by the semi-discrete scheme (4.14), which can be expressed, as $\frac{d}{dt}S = (\nabla_S^T)A_S$. On the other hand, the scheme for P can be written in the compact form

$$\frac{d}{dt}P = -\Big((\nabla_S^K)^T\big((\nabla_S^K)^{-T}\frac{d}{dt}S\big)(\nabla_S^K)\Big)P, \tag{4.34}$$

where $(\frac{d}{dt}S)(\nabla_S^K) \in \mathbb{R}^{N\times N}$ denotes the contraction of the discrete operator $(\nabla_S^K) \in \mathbb{R}^{D\times N\times N}$ and the matrix $(\frac{d}{dt}S) \in \mathbb{R}^{N\times D}$ in the first component and second component.

Proposition 4.2 (Semi-discrete algorithm for the Kolmogorov equation)**.** *Under the assumption* (4.22)*, consider $t \in [t_0,T] \mapsto S(t) \in \mathbb{R}^{N\times D}$ a solution to transported scheme* (4.14)*, with no drift, that is $r \equiv 0$, modeling a martingale process. Consider the scheme* (4.33)*.*

- **Consistency.** *The scheme* (4.33) *is consistent with a transported version of the Kolmogorov equation* (4.32)*. (Cf.* (4.41) *below.)*
- **Sup-norm estimate.** *The scheme* (4.33) *is global defined and enjoys the bound*

$$\sup_{s\in[0,t]}\sum_{n=1}^N (P^n(s))^2 \leq \sum_{n=1}^N (P^n(t))^2, \qquad t \in [0,T], \tag{4.35}$$

  (which is an analogue of the decay property of the moment $\int P^2 d\mu^{ex}$ for exact solutions).
- **Moment conservation property.** *The scheme* (4.33) *is conservative in the following sense:*

$$\sum_{n=1}^N P^n(t) = \sum_{n=1}^N P^n(T), \qquad t \in [t_0,T], \tag{4.36}$$

  (which is an analogue of the moment $\int P d\mu^{ex}$ for exact solutions).

- **Stochastic property.** *If* $\Pi = \Pi(s,t) \in \mathbb{R}^{N\times N}$ *denotes the forward generator of* (4.33)*, that is, the solution to*

$$\frac{d}{dt}\Pi(s,t) = B_S(t)\Pi(s,t), \qquad \Pi(s) = I_{N\times N}, \quad t \in [s,T], \tag{4.37}$$

then one has

$$P(s) = \Pi(s,t)P(t), \qquad s \in [t_0,t]. \tag{4.38}$$

The matrix $\Pi(s,t)$ *is a stochastic matrix provided*[1] $\nabla_S^T A_S \nabla_S$ *is essentially non-negative.*

We can view $\Pi(s,t)$ as a *transition probability matrix* between times s and t. We recall that an *essentially non-negative matrix* $M = (m_{ij})$ is a matrix with positive off-diagonal elements $m_{ij} \geq 0, i \neq j$. We recall also the following result (see [34]) : a matrix M is an essentially non-negative matrix if and only if $\exp(tM)$ is positive.

The stochastic property is an important property to check for in mathematical finance applications, and the associated scheme (4.33) satisfies a discrete maximum principle provided this condition is fulfilled. The condition $\big((\nabla_S)^T A(\nabla_S)\big)_{1\leq n,m\leq N} \geq 0$, for $n \neq m$, is a technical condition that restricts the possible shape of the basis functions.

Proof. Consistency of the approximation scheme. We consider the following weak formulation, for any solution P to the Kolmogorov equation (4.32) and any measure solution $\mu(t,\cdot)$ to the Fokker-Planck equation (4.21) without drift term (that is, $r \equiv 0$):

$$\frac{d}{dt} < \mu, P >_{<\mathcal{D}',\mathcal{D}>} = - < A\cdot\nabla^2 P, \mu >_{<\mathcal{D}',\mathcal{D}>} + < P, \nabla^2\cdot(A\mu) >_{<\mathcal{D}',\mathcal{D}>} = 0, \tag{4.39}$$

that is,

$$0 = \frac{d}{dt}\int P(t,\cdot)\mu(t,\cdot) = \frac{d}{dt}\int (P\circ S)(t,\cdot)dx. \tag{4.40}$$

Here, $S(t,\cdot)$ is a transport map of $\mu(t,\cdot)$, solving (4.30a) without drift ($r \equiv 0$). Thus we obtain

$$\frac{d}{dt}(P\circ S) = (\partial_t P)\circ S + \partial_t S(\nabla P)\circ S = (\partial_t P)\circ S + \nabla\cdot\Big((A\circ S)\mu\Big)(\nabla P)\circ S = 0. \tag{4.41}$$

Thus, as in (4.30a) $\nabla\cdot\Big((A\circ S)\mu\Big)$ is consistent with the operator $\nabla_S^T A_S$, and we conclude that the semi-discrete scheme

$$(\partial_t P)\circ S = -\nabla_S^T A_S \nabla_S P, \quad P \in \mathbb{R}^N$$

is consistent with the transported Kolmogorov equation. This semi-discrete scheme is mass-conservative, that is,

$$\frac{d}{dt} < P, 1_N >_{\ell^2} = -\langle A_S\nabla_S P, \nabla_S 1_N\rangle_{\ell^2} = 0,$$

as soon as $\nabla_S 1_N = 0$, that is fulfilled as soon as a polynomial basis is added.

Global existence. The energy bound

$$\frac{d}{dt}\|P\|_{\ell^2}^2 = -\langle A_S\nabla_S P, \nabla_S P\rangle_{\ell^2} \leq 0$$

as soon as the matrix field A_S is positive definite, hence leading to the estimate $\|P(s)\|_{\ell^2} \leq \|P(t)\|_{\ell^2}, \quad s \leq t$, and the scheme is a stable contracting scheme for the discrete ℓ^2 norm.

Stochastic property. A stochastic matrix $M \in \mathbb{R}^{N\times N}$ is a positive matrix $m_{i,j} \geq 0$, for which $\sum_j m_{i,j} = 1$, that is, the vector 1_N is an eigenvector associated with the eigenvalue 1. The same argument as the one for the mass conservation above applies:

$$\frac{d}{dt}\Pi 1_N = -\langle A_S\nabla_S\Pi, \nabla_S 1_N\rangle_{\ell^2} = 0.$$

[1] This condition can be ensured by restricting attention to a class of kernels, which should be in agreement with the dissipative aspects of the Kolmogorov operator.

The positivity $m_{i,j} \geq 0$ follows from (4.37), from which we deduce

$$\Pi \simeq \exp(\nabla_S A_S \nabla_S).$$

Provided the matrix $\nabla_S A_S \nabla_S)$ is essentially non-negative, it is positive. Obviously $\frac{d}{dt}\pi_{n,n} \leq 0$, $\pi_{n,n}(0) = 1$, but remains positive. □

4.5 Fully discrete, monotone, and entropy dissipative schemes

Reduction to a martingale process. We now present a full discretization of the Fokker-Planck equation (4.2) and the Kolmogorov equations (4.6), which are based on the semi-discrete schemes (4.14) and (4.17). We introduce a time discretization $0 = t^0 < t^\epsilon < t^1 \ldots < t^{N_t} = T$ with forward increments $\delta^n = t^{n+1} - t^n$. The discrete solutions are represented in a matrix form as $S^n \simeq S(t^n) \in \mathbb{R}^{N_S \times D}$, $P^n \simeq P(t^n) \in \mathbb{R}^{N_S \times D_P}$, $A^n \simeq A(t^n, S^n) \in \mathbb{R}^{N_S \times D \times D}$, etc.

Consider the scheme (4.14), and observe that it consists of a hyperbolic part and a (martingale) diffusive part, so at each time step t^n we split (4.14) as follows. The hyperbolic part is straightforwardly integrated with a classical ODE discretization and we denote by V^{n+1} the corresponding discrete solution produced at the time t^{n+1} which is an approximation to

$$\frac{d}{dt}V = r_V, \quad V^n = S^n, \quad t^n \leq t \leq t^{n+1}. \tag{4.42}$$

Once V^{n+1} is computed, the numerical scheme for the martingale part is written in the form (4.25), and is considered with the initial condition V^{n+1} at the time t^n while r_S can be replaced by zero, that is, we solve

$$\frac{d}{dt}S = B_S S, \quad B_S = -\big(\nabla_S\big)^T A_S \big(\nabla_S\big) \tag{4.43}$$

From now on, in order to present our fully discrete scheme we can thus focus on the martingale case without discussing further on the hyperbolic part.

Fully discrete algorithm. To compute the solution to (4.43), we first rely on a simple Euler scheme at an intermediate time $t^{n+1/2}$:

$$S^{n+1/2} = S^n + \sqrt{(1/2)\, A^n \delta^n}\, \epsilon_{\mathcal{N}}, \tag{4.44}$$

where we have the following properties.

- We recall that $A^n \in \mathbb{R}^{N_S \times D \times D}$ represents a field of symmetric positive definite matrix. The square root of $(1/2)A^n\delta^n$ is defined by taking the principal positive square root of each matrix, therefore $\sqrt{(1/2)\, A^n \delta^n} = \big(\sqrt{(1/2)\, A^n(t^n, S^n, S^n)\delta^n}\big)_{n=1,\ldots,N_s} \in \mathbb{R}^{N_S \times D \times D}$.
- $\epsilon_{\mathcal{N}} \in \mathbb{R}^{N_S \times D}$ is selected once for all as a given sampling of the D-dimensional normal law $\mathcal{N}^D(0,1)$, and we pick up $\epsilon_{\mathcal{N}}$ as an approximation of a sharp discrepancy sequence of $\mathcal{N}^D(0,1)$; see (4.14).
- $\sqrt{(1/2)\, A^n \delta^n}\, \epsilon_{\mathcal{N}}$ stands for the multiplication $\big(\sqrt{(1/2)\, A^n(t^n, S^n, S^n)\delta^n}\, \epsilon^n_{\mathcal{N}}\big)_{n=1,\ldots,N_s}$, which belongs to $\mathbb{R}^{N_S \times D}$ and thus is a vector field.

We now solve (4.43). Observe that a Crank Nicolson scheme for the approximation of this equation would produce, at time t^{n+1},

$$S^{n+1} = \big(\mathrm{Id} - \delta^n B_{S^{n+1/2}}\big)^{-1} \big(\mathrm{Id} + \delta^n B_{S^{n+1/2}}\big)\, V^n. \tag{4.45}$$

However, this approach will not be followed here since, although the Crank-Nicolson scheme enjoys nice properties in terms of entropy dissipation, it does not lead to a *monotone scheme*.

The matrix B_S is symmetric and we can work with few points N_S. We summarize our numerical scheme for the Fokker Planck equation as follows :

$$\begin{aligned} &S^{n+1/2} = S^{n+1} + \sqrt{\frac{A^n\delta^n}{2}}\epsilon_{\mathcal{N}}, \qquad S^{n+1} = \mathbf{\Pi}^{n,n+1} S^n \\ &\mathbf{\Pi}^{n,n+1} = \exp\big(\delta^n B^{n+1/2}\big), \qquad B^{n+1/2} = -\big(\nabla_{S^{n+1/2}}\big)^T A_{S^{n+1/2}} \big(\nabla_{S^{n+1/2}}\big). \end{aligned} \tag{4.46}$$

Once these quantities are computed for all $0 < t^0 < \ldots < t^{N_t} = T$, the backward Kolmogorov equation reduces to

$$P^{n+1} = \mathbf{\Pi}^{n,n+1} P^{n+1}. \tag{4.47}$$

Concerning the fully discrete scheme (4.46), our main result is (unsurprisingly) similar to Propositions 4.1 and 4.2.

Proposition 4.3 (Fully discrete algorithm of the coupled Fokker-Planck-Kolmogorov system)**.** *Consider a time discretization $t^0 < t^\epsilon < t^1 \ldots < t^{N_t} = T$ and the full time-discrete numerical scheme above, for the approximation of the transported Fokker-Planck equation* (4.2)*, under the ellipticity and linear growth conditions* (4.22)*. Then the scheme above defines a fully time-discrete family of moving particles $S^n = (S^n_{m,d})_{m,d} \in \mathbb{R}^{N_S \times D}$ together with an associated discrete measure $\mu_{S^n} = \frac{1}{N_S} \sum_m \delta_{S^n_m t}$, which determine a convergent approximation of the solution $\mu^{ex}(t^n)$ to* (4.21)*, as follows.*

- **Sup-norm estimate.** *The map $t \mapsto S(t)$ is defined for all times and remains globally bounded in the sup-norm, i.e.*

$$\sup_n e^{-\lambda t^n} \sum_{n=1}^N |S^n|^2 \leq \sum_{n=1}^N |S^0|^2. \tag{4.48}$$

- **Balance law of momentum.** *The sum of all components satisfies the evolution equation*

$$\langle S^n_d, 1_{N_S} \rangle_{\ell^2} = \langle S^0_d, 1_{N_S} \rangle_{\ell^2}, \quad d = 1, \ldots, D \tag{4.49}$$

  and, in particular, the total momentum is constant in time when the drift vanishes identically.

- **Error estimate.** *For any relevant test-function φ and any time $t \geq 0$ one has*

$$\left| \int_{\mathbb{R}^D} \varphi(x) d\mu^{ex}(t^n, x) - \frac{1}{N} \sum_{n=1}^N \varphi(S^n)) \right| \leq \mathbf{d}^K(\mu^{ex}(t^n), \mu_{S^n}) \, \|\varphi\|_{\mathcal{H}^K(\mathbb{R}^D)}, \tag{4.50}$$

  where the discrepancy error was defined in Section 2.

- **Stochastic property.** *The matrix $\mathbf{\Pi}^{n,n+1} \in \mathbb{R}^{N_S \times N_S}$, describing the transition probability $\pi_{l,m} = \mathbb{P}(S^{n+1}_m | S^n_l)$, is a stochastic matrix at each time t^n, which is also positive provided $(\nabla_S)^T A(\nabla_S)$ is positive (for all $n \neq m$).*

Proof. Consider S^{n+1} given by the scheme (4.46), with $0 \leq s \leq \delta^n$, together the expression $S^n(s) = \exp(sB^{n+1/2}) S^n$, implying that $S^n(\delta^n) = S^{n+1}$. We find

$$\frac{d}{ds} \|S^n(s)\|^2_{\ell^2} = \langle S^n(s), \nabla_S^T A_S \nabla_S S^n(s) \rangle_{\ell^2}$$

and so

$$\frac{d}{ds} \|S^n(s)\|^2_{\ell^2} = \langle \nabla_S S^n \exp\left(sB^{n+1/2}\right), A_S \nabla_S S^n \exp(sB^{n+1/2}) \rangle_{\ell^2} \tag{4.51}$$

Observe that $\nabla_{S^n} \exp\left(sB^{n+1/2}\right) S^n$ and

$$\frac{d}{ds} \|S^n(s)\|^2_{\ell^2} = \langle \nabla_S S^n \exp\left(sB^{n+1/2}\right), A_S \nabla_S S^n \exp\left(sB^{n+1/2}\right) \rangle_{\ell^2}. \quad \square$$

Algorithm for the transition probability matrix. It remains to approximate the transition probability matrix (4.11). Our algorithms is based on the following observation. If $t \mapsto X_t \in \mathbb{R}^D, t \geq 0$ is a martingale process, let us consider any number $N > 0$, any $0 \leq s \leq t$, and denote by $x \in \mathbb{R}^{N \times D}$ (resp. $y \in \mathbb{R}^{N \times D}$) any iid sample of X_t (resp. X_s). Then, the operator $\Pi(s, t, x, y)$ can approximated as the following stochastic matrix

$$\arg \inf_{\Pi \in \mathcal{S}_N} \|y - \Pi x\|_{\ell^2} = p(x|y) \in \mathbb{R}^{N \times N}, \tag{4.52}$$

where $\mathcal{S}_N$ denotes the set of $N \times N$ stochastic matrix. Our algorithm is described as follow.

Step 1. First average processes and denote : $y = y - \bar{y}$, $x = x - \bar{x}$, where $\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x^i$.

Step 2. We solve first

$$\epsilon = \arg\inf_{\epsilon\in\epsilon_N} \|y - \epsilon x\|_{\mathcal{H}^K},$$

where ϵ_N is the set of all permutations matrix. This amounts to consider the Linear Sum Assignment Problem (LSAP) problem :

$$\bar{\epsilon} = \arg\inf_{\epsilon\in\epsilon_N} d_k(\epsilon x, y),$$

where d_k is the pseudo-distance associated with the kernel. For the next step, we consider thus $\bar{\epsilon}x$ instead of x.

Step 3. We then solve

$$\arg\inf_{\Pi\in\mathcal{S}_N} \|y - \Pi x\|_{\ell^2},$$

where $\mathcal{S}_N$ is the set of all stochastic matrices. To this end, we consider a gradient descent method and compute

$$\frac{d}{dt}\|\Pi x - y\|^2_{\ell^2} = \big\langle \Pi x - y, \frac{d}{dt}\Pi x\big\rangle_{\ell^2}. \tag{4.53}$$

Observe that the previous equation does not provide a descent algorithm consistent with the stochastic matrix. However, (4.33)-(4.34) gives some indications to compute a stochastic matrix projection : we use the following algorithm to approximate the transition probability matrix

$$\begin{aligned}&\frac{d}{dt}\Pi = -\Big((\nabla_y)^T A(\nabla_y)\Big)\Pi, \qquad \Pi(0) = \mathrm{Id},\\ &a \text{ polar decomposition of the matrix field } (\nabla_y^T)^{-1}\big(\Pi x - y\big),\end{aligned}$$

where $(\nabla_y^T)^{-1}$ is a discrete operator defined from the kernel.

4.6 Application to business cases

Static hedging strategy. We now describe two particular applications of our method for finance problems, and we present first a static hedging strategy. Our setup will be slightly simplified and we refer to [26] for more general business cases. We thus consider a stochastic process X_t governed by the stochastic equation (4.1) corresponding, for definiteness, to a D-dimensional martingale log-normal process modeling forward rate swaps. Such a model can be taken to be the financial model of interest rates of Brace-Gatarek-Musiela [5], also referred to as the *LIBOR market model*. From the stochastic process X_t satisfying(4.1), we define a second stochastic process, referred to as the *underlying process* and, for definiteness, we choose the stochastic curves of the following forms:

- Stochastic discount curves $B(t, s, X_t)$ (defined for $s \geq t$), where $B > 0$ is a prescribed nonlinear function.
- Stochastic swap rates curves $r(t, s, X_t)$, defined by

$$r(t, s, X_t) = \frac{1 - B(t, s, X_t)}{B(t, s, X_t)(s - t)}. \tag{4.54}$$

The portfolio and hedging instruments are then described as follows. Given any instrument with payoff $P(t, s, X)$ (where t represents the initial time of the contract), we can consider its fair values $P(t, s, X_t)$ given by the Kolmogorov equation (4.6), or else consider its sensitivities $\nabla P(t, s, X_t)$ given by (4.19). These values are computed by the discretization scheme (4.17).

For instance, in [26] the Economic Value of Equity (EVE) and Net Income Interest (NII) indicators were chosen as instruments. They are written on top of each loans and assets of the bank account sheet, and they have both swaps rate curves and discount curves as underlyings. NII and EVE describe, with the simplest model, *perpetual swaps*. However, they can turn into barrier options (statistical models), or Bermudan swaptions (rational models), depending on how renegotiation and refund effects are taken into account.

Fair values and their sensitivities are themselves two stochastic processes and we can represent the numerical simulation as follows. In Figure (4.2), the left-hand figure represents the expectation of the EVE process as a function

of time, while the right-hand figure shows the sensitivities at time 1 with respect to one particular underlying, called LIB1Y in our model. We considered here the EVE indicator together with a statistical re-negociation model.

In this setting, a hedge instrument is also a derivative, having payoff $H_m(t, s, X)$, inside an hedging portfolio of M instruments $H = (H_1, \ldots, H_M)$. For instance we considered swaps, caps and floors as hedging instruments in [26], but swaptions, even Bermudan ones could be included.

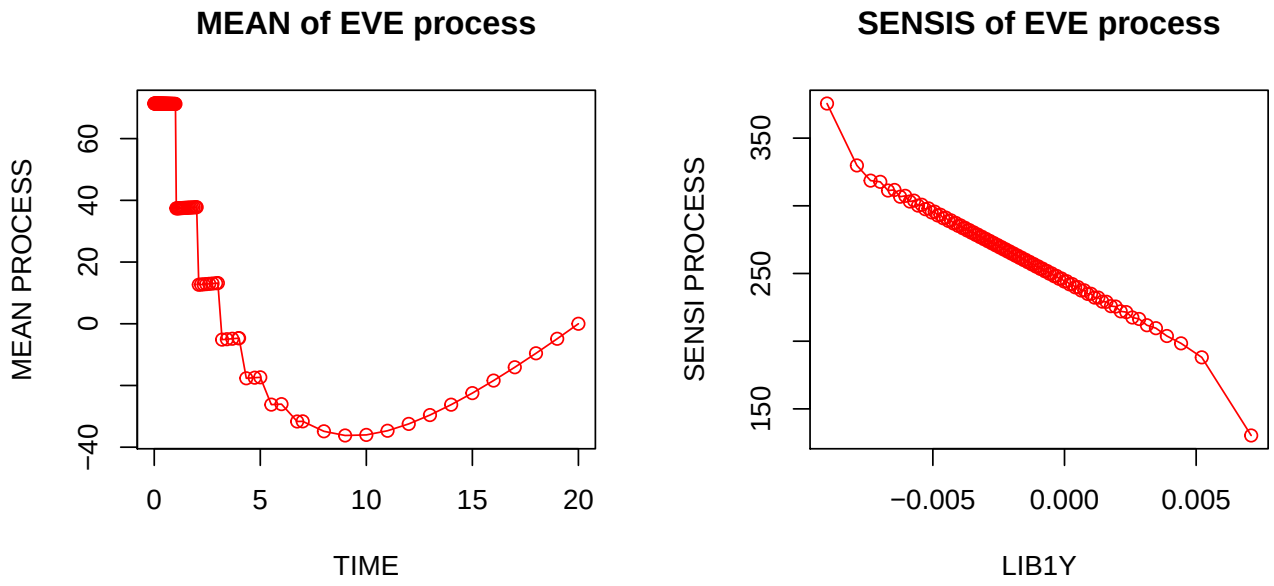


Figure 4.2: Expectation (left-hand side) and sensitivity (right-hand side) at $t = 1$

Hedging strategies. A *static* hedging strategy, set at the time value t, is any function $\alpha(t) = (\alpha_1, \ldots, \alpha_M)(t) \in \mathbb{R}^M$, defining a portfolio with payoff $H_\alpha =< \alpha, H >$. We can compute any strategy associated with an objective criteria such as

$$\overline{\alpha} = \arg \inf_{<\alpha, \overline{H}> \leq C} \int_{s \geq t} I\big([P - < \alpha, \overline{H} >](t, s, \cdot)\big) ds, \tag{4.55}$$

in which (referring to [26] for further details):

- I is a convex functional and, for instance, considered the variance $\int_{\mathbb{R}^D} (P - P_{mean})^2 d\mu$, or the sensitivities $\int_{\mathbb{R}^D} \left|\nabla P\right|^2 d\mu$.
- $< \alpha, \overline{H} > \leq C$ might be added or not. The purpose of this constraint is to limit the hedging portfolio investment value.

Observe that the hedging strategy (4.55) is based on fair values, which are computed in the course of solving the Kolmogorov equation. On the other hand, with a standard Monte-Carlo approach it would be very costly to compute such hedges.

In Figure 4.3 we plot the numerical results for such a hedging strategy. The left-hand figure shows the variance of the EVE over time, represented in Figure 4.2 in blue color, while the red color shows the variance of the EVE after hedging its sensitivities using (4.55). This hedge reduces to almost zero the variance or the sensitivity of the EVE process. However, it also increases dramatically the ones of the NII process; see the right-hand figure.

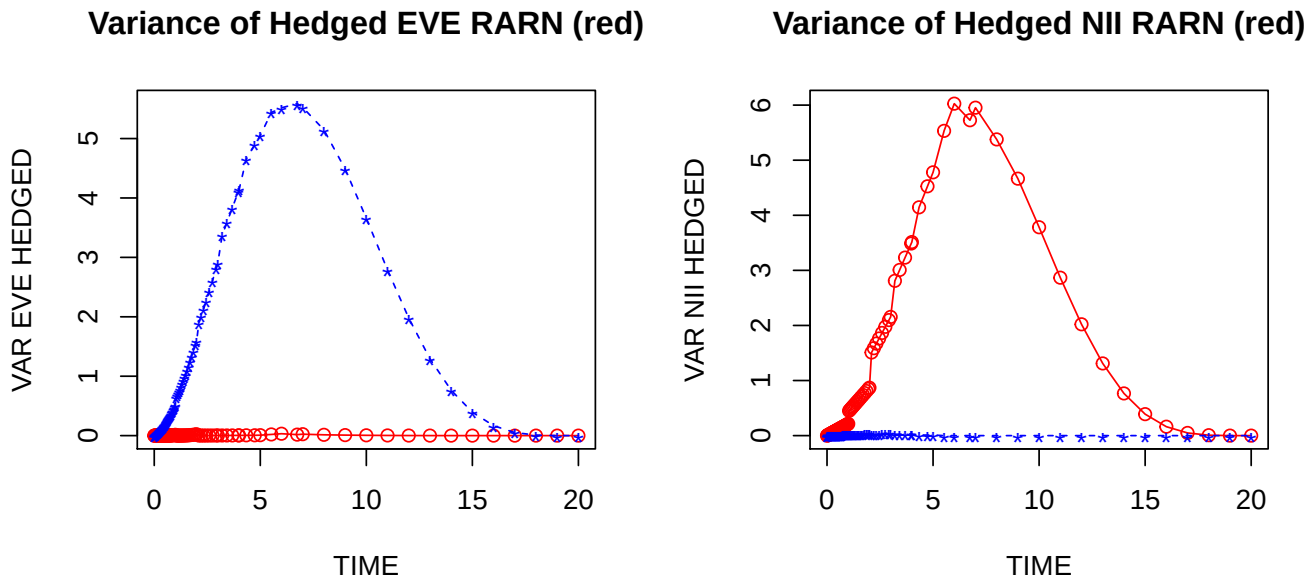


Figure 4.3: left-expect. EVE right-sensi. EVE t=1

β-Hedging strategies. There are several cases of interest where one would like to mix two different hedging strategies. For instance, in the NII / EVE example treated so far, we are using two different hedges computed with (4.55). In the first one, we compute the hedging strategy minimizing the EVE sensitivities, as illustrated in Figure 4.3, while in the second one we compute an hedge aiming to reduce the NII sensitivities. This second strategy is very efficient to allows one to lower the NII sensitivities or variance but, as shown in Figure 4.3, it increases dramatically the ones of the EVE.

Now, given any two strategies $\overline{\alpha}^1, \overline{\alpha}^2$ computed with (4.55), let us consider the strategy

$$\alpha = \beta\overline{\alpha}^1 + (1-\beta)\overline{\alpha}^2, \quad 0 \leq \beta \leq 1,$$

where β is chosen to match another criteria. For instance, we picked up in Figure (4.4) $\beta = 0.07$. The figure plots in blue the variance of the NII and EVE in blue. The variance resulting form the β-hedge is plot with red color, showing a reduction for both EVE and NII.

The value $\beta = 0.07$ in this example is chosen in order to keep the variance of the EVE inside a given limit, that can be of internal bank or regulatory nature. Within this limit, we lowered the sensitivities of the NII as much as possible.

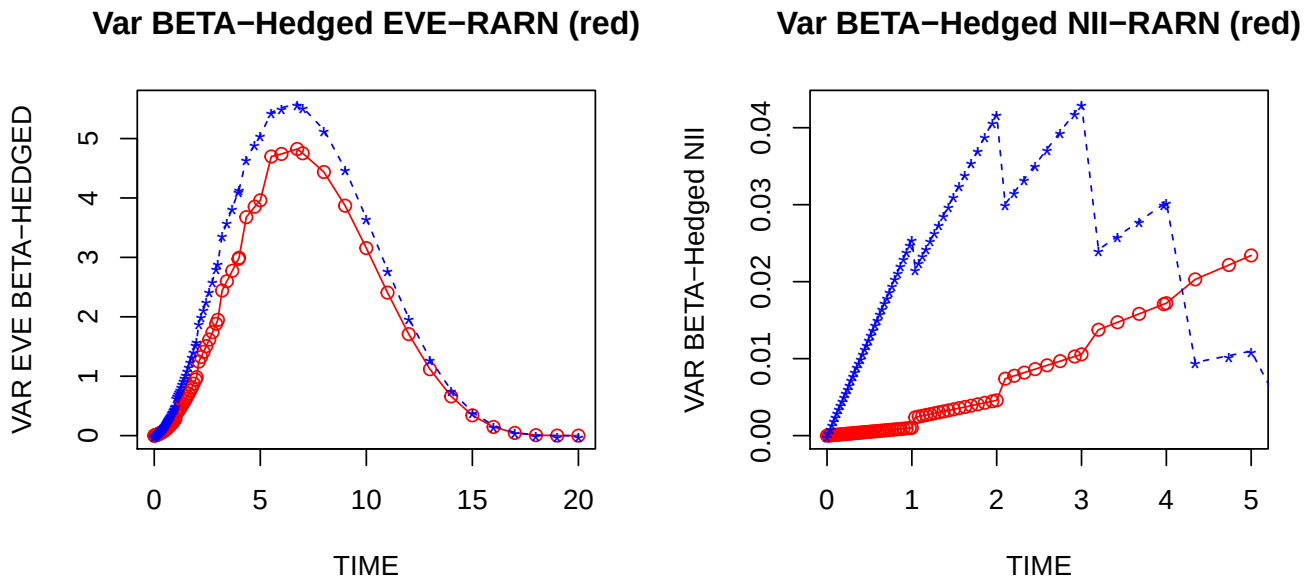


Figure 4.4: β-hedge, $\beta = 0.07$

Application to metrics for autocalls. We finally describe business case which illustrate the performance achieved by our transport mesh-free method for finance applications. The business case consists in computing metrics for big portfolio of Autocalls. The motivation is a pre-sale equity-derivative front-office case. It allows us to perform a pre-selection for customers of such instruments. The financial setting is the following:

- Consider a D dimensional stochastic process (4.1) adapted to share modeling, for instance a log-normal one.
- Build the underlying process, modeling a basket of shares with a stochastic process. To model each share, we use the Buehler dividend model [6] together with the local volatility calibration algorithm, as first proposed in [15].
- Consider a template Autocall with some given characteristic and consider as its underlyings any combination of our D shares. There are exactly 2^D combinations possible of shares as underlyings to our Autocall. On each, we compute various metrics as prices, deltas, gammas, coupon values, etc. These metrics are then compiled and mixed to match clients needs.

To evaluate the accuracy and performance of our method, we performed the following tests.

- We considered a one-dimensional stochastic process, that is a log-normal martingale process, with 10 % volatility.
- We considered a D-dimensional process copying D times the previous one, without correlation.

Next, consider the portfolio consisting of 2^D Autocalls defined as any combination of underlyings. Since each underlyings are copies, with this construction we only have D different Autocalls, each one depending only upon its number of underlyings. Thus we can quite easily provide a benchmark of our metrics. For instance, in Figure 4.5 we present a benchmark for our computed prices for $D = 10$. The figure shows the numerical results using the 10 theoretical prices on the horizontal axis, while the vertical axis is used for the 1024 prices calculated by our method. Importantly, the computation times are below a minute in dimensions $D = 10, 11, 12, 13$, using a single core of an Intel I7 processor.

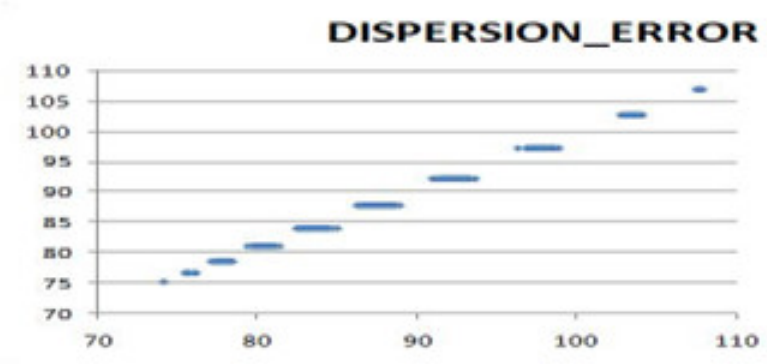


Figure 4.5: dispersion error, $D = 10, N = 512$

5 An algorithm for the signed-polar decomposition in transport theory

5.1 Polar and signed-polar decompositions in finite dimensions

Two decompositions of complex numbers. The polar decomposition is an important tool in many applications, and we will now present a discrete algorithm for its computation. We begin by a discussion at the continuous level and present several (old and new) notions which allow one to decompose matrices and mappings, and will play a central role throughout. The polar decomposition of a complex number $z = \rho e^{i\theta} \neq 0$ can be revisited as follow: While it is standard to impose the normalization $\rho > 0$ and $\theta \in [0, 2\pi)$, we propose here an alternative decomposition.

Definition 5.1. *The* signed-polar decomposition *of a complex number $z = \rho e^{i\theta} \neq 0$, by definition, is $z = \rho' e^{i\theta'}$ in which one relaxes the signed condition on the modulus while the angle is restricted to remain in the "half-space", i.e. $\rho' = \pm\rho$ and $\theta' \in (-\pi/2, \pi/2]$.*

In other word, from the standard polar decomposition $z = \rho e^{i\theta}$, we set

$$(\rho', \theta') = \begin{cases} (\rho, \theta), & \theta \in [0, \pi/2), \\ (-\rho, \theta - \pi), & \theta \in [\pi/2, 3\pi/2), \\ (\rho, \theta - 2\pi), & \theta \in [3\pi/2, 2\pi). \end{cases} \tag{5.1}$$

Observe that (ρ, θ) can be characterized by the property that θ is a solution to the following minimization problem

$$\inf_{\theta \in [0,2\pi)} |z - e^{i\theta}|^2 = \inf_{\theta \in [0,2\pi)} |1 - e^{-i\theta} z|^2, \tag{5.2}$$

while (ρ', θ') can be characterized by the minimization problem

$$\inf_{\theta \in (-\pi/2, \pi/2]} |z - e^{i\theta}|^2 = \inf_{\epsilon = \pm 1} \inf_{\theta \in (-\pi/2, \pi/2]} |\epsilon - e^{-i\theta} z|^2. \tag{5.3}$$

In the polar decomposition, one performs a projection of z on the unit circle in the complex plane, while in the signed-polar decomposition, we perform a projection on the unit half-circle of numbers with positive real part. We also observe that the mapping $z \neq 0 \mapsto (\rho, \theta)$ is continuous (and even analytic) *except* across the positive real line, while the mapping $z \neq 0 \mapsto (\rho', \theta')$ is continuous (and even analytic) *except* across the imaginary line. Interestingly, such a discontinuity property does not arise with real-valued matrices, nor mappings, as we show below. Furthermore, it will convenient to slightly modify the above notation. In the following, in dealing with matrices and maps, we will extend the following notation stated here for complex numbers:

$$\begin{aligned} &z = \rho^+ e^{i\theta}, && \rho^+ > 0, && \theta \in [0, 2\pi), \\ &z = \rho e^{i\theta^+}, && \rho \in \mathbb{R}, && \theta^+ \in (-\pi/2, \pi/2]. \end{aligned} \tag{5.4}$$

Two decompositions of real-valued matrices. We next consider matrices and throughout we deal with real-valued $N \times N$ matrices. Recall that the standard *polar decomposition* of an invertible matrix M reads

$$M = US^+, \qquad U \text{ is orthogonal}, \quad S^+ \text{ is symmetric and positive definite.} \tag{5.5}$$

Observe that the map $M \mapsto (U, S^+)$ is well-defined on the set of invertible matrices, while the decomposition may fail to be unique for non-invertible matrices. In fact, when M is not invertible, its symmetric part $S^+ = (M^T M)^{1/2}$ is defined uniquely as a semi-positive definite matrix, so that uniqueness fails only for the othogonal part U.

In contrast with the decomposition above, in the definition we now propose, the positivity condition is imposed on the orthogonal part of the decomposition rather than on the symmetric part. Clearly, the space of symmetric orthogonal matrices clearly plays a particular role here.

Proposition 5.2 (The signed-polar decomposition of matrices)**.** *1. Any (invertible) matrix M admits a decomposition of the form*

$$M = U^+ S, \qquad U^+ \text{ is orthogonal and positive definite}, \quad S \text{ is symmetric.} \tag{5.6}$$

referred to as its signed-polar decomposition. *Moreover, the mapping $M \mapsto (U^+, S)$ is continuous when restricted to the space of invertible matrices.*

2. The connection between the polar and signed-polar decompositions $M = U^+S = US^+$ of an invertible matrix is as follows: there exists a symmetric and orthogonal *matrix denoted by U_0 such that*

$$U^+ = UU_0, \qquad S = U_0 S^+. \tag{5.7}$$

Furthermore, if $S = VDV^T$ denotes the spectral decomposition of S in which V consists of an orthogonal basis of eigenvectors and D is a diagonal matrix made of its real eigenvalues, then one has

$$S^+ = VD^+V^T, \qquad U_0 = V\Sigma V^T, \tag{5.8}$$

where D^+ is defined by taking the absolute value of the elements of D and Σ is a diagonal matrix made of the sign of these eigenvalues.

Proof. Any invertible matrix M admits a complex-valued, singular value decomposition $M = WD^+V^*$, where W, V are unitary matrices and the diagonal matrix D^+ is *real and positive.* Here, V^* stands for the adjoint of the matrix V. The standard polar decomposition is obtained immediately by writing

$$M = (WV^*)(VD^+V^*) = US^+.$$

For any diagonal matrices Σ having ± 1 on its diagonal, we have

$$M = (W\Sigma V^*)(V\Sigma D^+V^*) = \big(UU_0\big)\big(U_0^* S^+\big), \quad U_0 = V\Sigma V^*$$

Observe that for any $x \in \mathbb{R}^N$ we have $< (W\Sigma V^*)x, x >=< \Sigma y, W^*Vy >, \quad y = V^*x$ Moreover, any unitary matrix is diagonalizable with orthogonal eigenspaces. In particular if one pick-up Σ carrying the signs of W^*V, then the previous expression is positive, leading to the signed-polar decomposition: $M = (W\Sigma V^*)(V\Sigma DV^*) = U^+S$. □

5.2 Polar and signed-polar decompositions of mappings

Notation. Let us repeat some of our notation. We write $\nabla f = \big(\partial_d f\big)_{d=1,\dots,D}$ for the gradient of a scalar-valued function $f : \mathbb{R}^D \mapsto \mathbb{R}$, and we denote by $\nabla \cdot S$ the divergence operator of a map $S : \mathbb{R}^D \mapsto \mathbb{R}^D$. The Laplace operator is expressed as $\Delta = \sum_{i=1,\dots,D} \partial_i^2 = \nabla \cdot \nabla$, and the Jacobian is writen as $\nabla S = (\partial_j S_i)_{i,j}$ (row-oriented). We also use the composition rule $\nabla(\varphi \circ S) = (\nabla S)(\nabla \varphi) \circ S$.

We can work on any open and convex subset $\Lambda \subset \mathbb{R}^D$ endowed with the normalized Lebesgue measure m, with $m(\Lambda) = 1$, but in practice we can simply pick up the unit cube $\Lambda = (0,1)^D$, and this is what we do in all our numerical tests. Consider maps $S : y \in \Lambda \mapsto x = S(y) \in \Omega$ taking values in a convex and open subset $\Omega \subset \mathbb{R}^D$. Given a probability measure μ defined on Ω and a probability measure ν defined Λ, a map $S : (\Lambda, \nu) \mapsto (\Omega, \mu)$ is said to be measure-preserving, or to transport ν into μ, provided $S_\sharp \nu = \mu$. In other words, one requires the following change of variable formula $\int_\Omega \varphi \, \mu = \int_\Lambda (\varphi \circ S) \, \nu$. for all test-functions φ. Under suitable regularity on the map S, this change of variable implies the Jacobian equation $(\mu \circ S) \, |\det \nabla S| = \nu$.

The standard polar factorization for mappings. By the theory of optimal transport [35], given a positive probability measure μ defined on Ω, (satisfying supp $\mu = \Omega$ and absolutely continuous with respect to the Lebesgue measure) there exists a unique *optimal transport map* $S : (\Lambda, m) \mapsto (\Omega, \mu)$ transporting the Lebesgue measure m on Λ into μ. It is uniquely characterized by minimizing a suitable "cost" functional. Precisely, to any map $S : \Lambda \to \Omega$ and probability measure $\mu = S_{\#}m$, we can associate the decomposition

$$S = (\nabla h^+) \circ T, \qquad h^+ : \Lambda \to \mathbb{R} \text{ convex}, \qquad T_{\#}m = m, \tag{5.9}$$

which is called the *polar factorization* of S. Hence, any map can be regarded as a gradient of a convex function ∇h^+, modulo a Lebesgue measure-preserving map T. As for the polar decomposition of matrices, let us consider the application $S \mapsto (h^+, T)$ which is defined whenever S is invertible in the sense that supp $S_{\#}m = \Omega$ and $S_{\#}m \leq C\, m$. The so-called Wasserstein distance between two probability measures μ_0 on Ω and μ_1 on Λ

$$d\big(\mu_0, \mu_1\big) = \mathbf{W}_2(\mu_0, \mu_1\big) = \inf_{S_{\#}\mu_1 = \mu_0} \int_{\Omega} |X - S(X)|_2^2 \, d\mu_0 \tag{5.10}$$

allows one to characterize the polar factorization $S = \nabla h : \Lambda \mapsto \Omega$ by solving a minimization problem.

The signed-polar factorization for mappings. As for matrices, in the following proposition, we propose here a novel polar factorization for mappings $S = \nabla h \circ T$, in which the positivity condition is imposed here on the part T rather than on the convex function h.

Proposition 5.3 (Signed polar factorization for maps)**.** *Let Λ be a convex set and $S : \Lambda \to \mathbb{R}^D$ be a map satisfying supp $S_{\#}m = \mathbb{R}^D$ and $S_{\#}m \lesssim m$. Then there exists a unique Lebesgue-measure-preserving map T^+ and a real-valued $h : \Lambda \to \mathbb{R}$ (which need not be convex) such that*

$$S = \nabla h \circ T^+, \quad T^+_{\#}m = m, \qquad \nabla T^+ \geq 0. \tag{5.11}$$

A connection between signed and polar decomposition is as follows: let $S = (\nabla h^+) \circ T$, h^+ convex, $T_{\#}m = m$ the standard polar factorization. Then there exists a Lebesgue measure-preserving $T_0 = \nabla g$ such that

$$T = T_0 \circ T^+, \qquad \nabla h = \nabla h^+ \circ T_0. \tag{5.12}$$

Of course, the signed-polar decomposition is trivial in dimension $D = 1$, while the standard polar factorization is non-trivial, corresponding to a reordering by increasing values, and generates Lebesgue-measure preserving field T that are *non-smooth re-arrangement* of the function. The regularity property of our factorization may be viewed as an important advantage in some applications. For instance, consider a map $S : \Lambda \mapsto \mathbb{R}^N$ with bounded total variation in the sense that its Jacobian is a measure-valued field of matrices. Then the signed-polar decomposition allows to give a meaning to $\nabla S = (\nabla T^+)(\nabla^2 h) \circ T^+$. while the formula $\nabla S = \nabla T (\nabla^2 h^+) \circ T$ for the standard polar decomposition is not well-defined.

Proof. Given an arbitrary mapping $S : \Lambda \mapsto \mathbb{R}^D$, let us consider its Jacobian ∇S. Consider the polar and signed-polar decomposition of matrices, say $\nabla S = U^+ W = U W^+$, where W defined on Λ is a square-integrable diagonal field of matrix, and U^+ is an orthogonal positive field of matrix, hence is bounded. Let us introduce the following equations

$$\nabla T^+ = U^+, \qquad \nabla P = W \circ (T^+)^{-1} \tag{5.13}$$

Since the polar decomposition is unique for any field $S = (\nabla h) \circ T$, the equation $\nabla T^+ = U^+$ has a solution for any field S with positive Jacobian, which can be expressed as (cf. the Hodge decomposition in Section 5.3): $T^+ = \Delta^{-1} \nabla \cdot U^+$. Furthermore, T^+ is Lipschitz continuous since ∇T^+ is orthogonal and positive. Observe that $|\det \nabla T^+| = 1$, hence we deduce $T : \Lambda \mapsto \Lambda$ is Lebesgue-measure-preserving, $T_{\#}m = m$, and the composition $S \circ (T^+)^{-1}$ is meaningful. In particular, we can solve the second equation as previously, computing explicitely

$$P = \Delta^{-1} \nabla \cdot \Big(W \circ (T^+)^{-1} \Big).$$

In particular, since ∇P is a symmetric field of matrix, we must have $P = \nabla h$, for some $h : \Lambda \to \mathbb{R}$. Finally, we check

$$\nabla((\nabla h) \circ T^+) = \nabla T^+ (\nabla^2 h) \circ T^+ = U^+ W = \nabla S,$$

hence, up to a constant, $P \circ T^+ = S$.

Consider now the connection between polar and signed-polar factorizations, that is, $S = \nabla h \circ T^+ = \nabla h^+ \circ T$. We deduce $T_0 = T \circ (T^+)^{-1} = (\nabla h^+)^{-1} \circ \nabla h$ is Lebesgue measure-preserving, and has a symmetric Jacobian field of matrices. Using the same argument as above, we conclude that $T_0 = \nabla g$. □

5.3 A continuous algorithm for computing polar decompositions

The Hodge decomposition. Here, $\Lambda = [0,1]^D$ is chosen to be the unit cube (but our analysis would hold for any convex subset of $\mathbb{R}^D$) and we denote its boundary by Γ. The *Hodge* decomposition consists in decomposing a map $S : \Lambda \to \mathbb{R}^D$ as an (for the L^2 norm) orthogonal sum of a gradient map plus a divergence-free map. More precisely, there exists a unique harmonic function h on Λ, a divergence-free map ζ on Λ and a function h_0 vanishing on the boundary Γ such that

$$S = \nabla(h + h_0) + \zeta, \qquad \nabla \cdot \zeta = 0. \qquad \zeta \cdot \eta = 0, \qquad \Delta h = 0, \tag{5.14}$$

where η is the (outward) normal to Γ and Δ^{-1} is the inverse Laplacian operator with vanishing boundary conditions. Then the components are determined (up to a constant for h_0) as

$$\nabla h_0 = \Pi S = \nabla \Delta^{-1} \nabla \cdot S, \qquad \zeta = \mathcal{L} S = \big(\mathrm{Id} - \nabla \Delta^{-1} \nabla \cdot\big) S, \tag{5.15}$$

where $\mathcal{L}$ is referred to as as the *Leray operator.*

Inverse and composition of maps. A mapping $S = S(t, \cdot)$, viewed as an input data, is a priori given (or computed in the applications). In particular, it might not define a proper mapping in the sense that $\det \nabla S > 0$ might not hold. Consider the following equation with unknown $u : \mathbb{R}^d \mapsto \mathbb{R}$ and prescribed data v

$$u \circ S = v. \tag{5.16}$$

Consider any map S and its polar factorization (see (5.9)), say $S = (\nabla h) \circ T$, with a convex function h and a surjective map $T \in \mathcal{A}(\Lambda)$, with $S(\Lambda) = \Omega$. Then the following identity

$$S^{-1} = T^{-1} \circ (\nabla h)^{-1} : \mathbb{R}^D \mapsto \Lambda \tag{5.17}$$

makes sense provided $\mu = S_\# m$ is dominated by the Lebesgue measure, and this leads us to the following solution of (5.16):

$$u = v \circ T^{-1} \circ (\nabla h)^{-1}. \tag{5.18}$$

To cope with the possibility that μ has a singular part with respect to the Lebesgue measure m, we should use a suitably generalized inverse.

Lebesgue measure-preserving maps. The following lemma shows that any path of divergence free vector generates a time-dependent Lebesgue measure-preserving maps.

Lemma 5.4 (Time-dependent Lebesgue measure-preserving maps)**.** *Consider a time-dependent path of Lebesgue-preserving map $T = T(t, \cdot)$ defined on Λ with $T(t, \cdot)_\# m = m$. Then there exists a time-dependent divergence-free map $\chi = \chi(t, \cdot)$ on Λ such that*

$$\partial_t T = \chi \circ T. \tag{5.19}$$

Proof. If $T(t, \cdot)_\# m = m$ is Lebesgue-preserving, it is a one-to-one map from Λ into itself and we can define $T^{-1}(t, \cdot)$ pointwise. For any smooth function φ, using the map $y = T(t, z)$, we have $\int_\Lambda \varphi \circ T = \int_\Lambda \varphi$, and we find

$$0 = \frac{d}{dt} \int_\Lambda \varphi = \frac{d}{dt} \int_\Lambda \varphi \circ T = \int_\Lambda (\nabla \varphi) \circ T \cdot \partial_t T = \int_\Lambda (\nabla \varphi) \cdot (\partial_t T) \circ T^{-1},$$

showing that $(\partial_t T) \circ T^{-1}$ is divergence-free. Conversely, assuming T satisfying (5.19), then it is now clear by the same computation that T is Lebesgue measure-preserving. □

A direct algorithm for polar factorizations. We arrive at the formulation of our continuous method for computing the polar factorization, using a connection between factorization of matrix and maps.

Proposition 5.5 (Factorization of smooth positive maps)**.** *Let $S : \Lambda \to \mathbb{R}^D$ be smooth map with convex support supp $S_\# m$ and positive Jacobian $\nabla S > 0$. Consider its polar factorization $S = (\nabla h) \circ T$. Then, up to a constant, one has*

$$T = \Delta^{-1} \nabla \cdot U, \qquad \nabla h = \Delta^{-1} \nabla \cdot (P \circ T^{-1}), \tag{5.20}$$

where $(P, U^+) : \Lambda \mapsto \mathcal{M}(\mathbb{R}^{D\times D})^2$ is the field of symmetric and orthogonal matrices generated by the polar decomposition of matrices $\nabla S = UP$.

Proof. We start from the polar factorization $S = (\nabla h) \circ T$, thus $\nabla S = \nabla T (\nabla^2 h) \circ T$. Cconsider the Jacobian ∇S and its polar decomposition in (5.5), thus $\nabla S = UP$, where P is a square-integrable field of symmetric positive matrix and U is an orthogonal positive field of matrix. Since the polar factorization $S = (\nabla h) \circ T$ is unique, we can identify both parts as follows:

$$\nabla T = U, \qquad \nabla^2 h = D \circ T^{-1}. \tag{5.21}$$

Consider the equation $\nabla T = U$ and recall that the Hodge decomposition yields the unique candidate to be a solution: $T = \Delta^{-1} \nabla \cdot U$, which is Lipschitz continuous. Since T is unitary, we can consider its inverse T^{-1} and the composition $D \circ T^{-1}$ is meaningful. In particular, we can solve the second equation as previously, computing explicitely

$$\nabla h = \Delta^{-1} \nabla \cdot \big(D \circ T^{-1}\big),$$

leading to a solution $h : \Lambda \to \mathbb{R}$ Finally, we check that $\nabla((\nabla h) \circ T) = \nabla T (\nabla^2 h) \circ T = UD = \nabla S$, hence, up to a constant, $P \circ T = S$. □

A steepest descent algorithm for polar factorizations. Proposition 5.5 now suggests a direct method in order to compute the polar factorization. As for polar factorization of matrices, we state a steepest descent iterative algorithm, as follows.

Proposition 5.6 (Polar factorization of smooth convex maps via steepest descent)**.** *Given any smooth map S with positive Jacobian matrix $\nabla S \geq 0$, consider the minimization problem:*

$$\overline{T} = \arg \min_{T_\# m = m} \int_\Lambda |\mathcal{L}(S \circ T)|^2 \, m, \tag{5.22}$$

which amounts to find the polar and the signed-polar factorizations $S \circ \overline{T} = \nabla h$, as defined in (5.9)*. The steepest descent algorithm for this minimization problem amounts to compute a family of Lebesgue-preserving maps $t \mapsto T(t, \cdot)$ with positive Jacobian $\nabla T \geq 0$ satisfying*

$$\begin{aligned} &S \circ T = \nabla h + \zeta, \qquad \nabla \cdot \zeta = 0 \\ &\partial_t T^{-1} = \mathcal{L}\big(\nabla^2 h \zeta\big) \circ T^{-1}, \qquad \partial_t T = -(\nabla T) \mathcal{L}\big(\nabla^2 h \zeta\big), \end{aligned} \tag{5.23}$$

where $\big(h, \zeta\big)(t, \cdot)$ is obtained via the Hodge decomposition (5.14) *in Λ, and $T^{-1}(t, \cdot)$ denotes the inverse of T and $\mathcal{L}$ the Leray operator. When supp $S = \Omega$, consider a path $T = T(t, \cdot)$ of Lebesgue measure-preserving maps satisfying* (5.23)*. Then, as $t \to +\infty$, the map $t \to S \circ T(t, \cdot)$ converges strongly at an exponential rate toward the signed-polar factorization of $\overline{S}$:*

$$S \circ T(t, \cdot) \to \overline{S} = \nabla \phi, \qquad \phi : \Lambda \to \mathbb{R} \text{ is convex, as } t \to +\infty. \tag{5.24}$$

For convex maps, the choice $\partial_t T^{-1} = \zeta \circ T^{-1}$ is an alternate and simpler computational choice.

Proof. Let us write $\partial_t T^{-1} \circ T = \chi$, where by Lemma 5.4 $\chi(t, \cdot)$ is a divergence-free map. Let us consider the Hodge decomposition $S \circ T = \nabla h + \zeta$, where $\zeta(t, \cdot)$ is divergence free. Then we have

$$\begin{aligned} \frac{d}{dt} \int_\Lambda |\zeta|_2^2 = \frac{d}{dt} \int_\Lambda |S \circ T - \nabla h|_2^2 = \frac{d}{dt} \int_\Lambda |S - \nabla h \circ T^{-1}|_2^2 \\ = \int_\Lambda < \nabla h \circ T^{-1} - S \cdot (\nabla^2 h) \circ T^{-1} \partial_t T^{-1} > . \end{aligned}$$

Composing back with T, we obtain

$$\int_\Lambda < \nabla h - S \circ T \cdot (\nabla^2 h)(\partial_t T^{-1}) \circ T >= -\int_\Lambda < (\nabla^2 h)\zeta \cdot \chi >, \tag{5.25}$$

thus the steepest descent algorithm reduces to $\chi(t,\cdot) = \mathcal{L}\big((\nabla^2 h)\zeta(t,\cdot)\big)$. □

5.4 A discrete algorithm for computing polar decompositions

Computing the polar factorization. We now describe our discrete algorithm which generates a polar factorization, along the guidelines provided by our theoretical study. From the numerical standpoint, our polar factorization algorithm takes as an input any two distributions of equal size $x \in \mathbb{R}^{N\times D}$, $y \in \mathbb{R}^{N\times D}$ and generates, as an output, $z \in \mathbb{R}^{N\times D}$, obtained by solving the discrete counterpart to (5.24)

$$y = \nabla_z h \quad (h,z) = \arg \min_{z\in\mathbb{R}^{N\times D}, h\in\mathbb{R}^N} \|y - \nabla_z h\|^2_{\ell^2} \tag{5.26}$$

This minimization problem relies on the Helmholtz-Hodge decomposition, which is computed using a kernel (cf. [18] for further details). We emphasize that this algorithm assumes that the initial map is positive in the sense that its Jacobian is positive, that is, $\nabla_x h \geq 0$. To this purpose, a reordering algorithm is required (as described in detail in [20]). In particular, the operator $(\nabla_x)^{-1}$ is described in [18].

Algorithm 1 Compute $y = \nabla_x h$ with a convex h.

Require: An admissible kernel k, a threshold parameter $\epsilon > 0$, two distributions of points $x \in \mathbb{R}^{N\times D}$, $y \in \mathbb{R}^{N\times D}$.
– update $x \leftarrow x^\sigma$, where $\sigma = lsap(d_k(x,y))$ is the permutation computed by the linear sum assignment problem [20] and d_k is the discrepancy error matrix.
while $\|y - \nabla_x h\|_{\ell^2} < \epsilon$ **do**
– compute the Helmholtz-Hodge decomposition $h = (\nabla_x)^{-1}(y) \in \mathbb{R}^N$ and $\zeta = y - \nabla_x h \in \mathbb{R}^{N\times D}$.
– compute $\lambda = \arg\min_{\lambda>0} \|y - (\nabla_x h)(w)\|^2_{\ell^2}$ and $w = x - \lambda\zeta$.
– update $x \leftarrow x - \lambda\zeta$.
end while
– output x, h.

The sampling algorithm. We illustrate the use of the polar factorization by presenting a function allowing to recover, from any input distribution $x \in \mathbb{R}^{N_x\times D}$ and any integer $M > 0$, a new distribution $z \in \mathbb{R}^{M\times D}$ which has very similar statistical properties to the initial distribution x. We can use this sampling algorithm in a number of practical situations, as it allows us to produce test data in order to check other algorithms. This algorithm is described as follows.

Algorithm 2 Compute M iid sample from any input distribution.

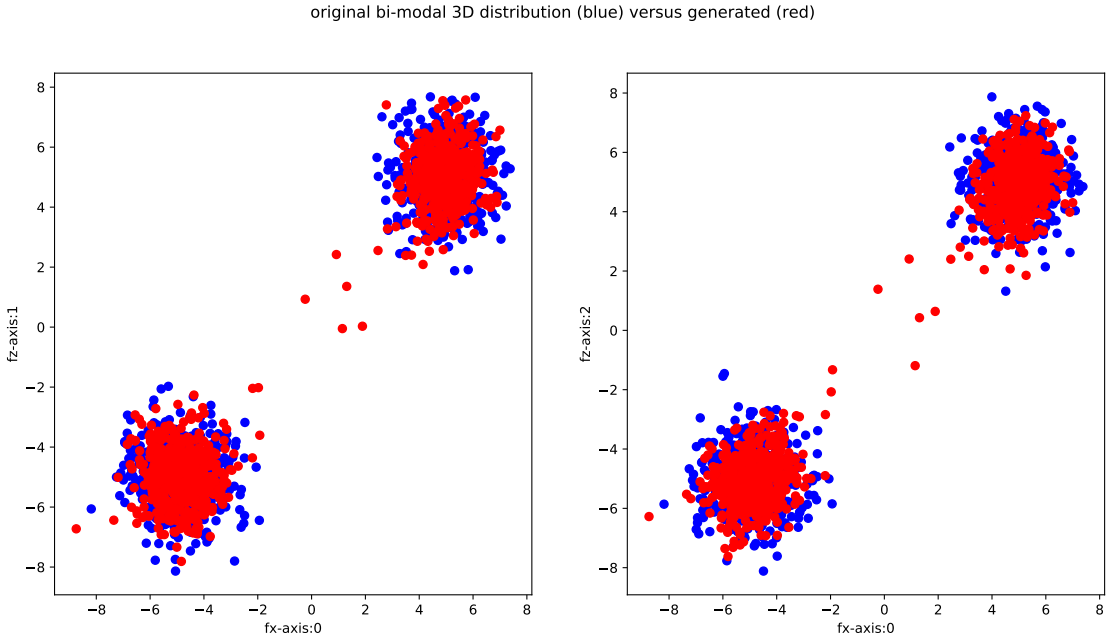
Require: k a positive definite kernel, $\epsilon > 0$ a threshold, a distributions $y \in \mathbb{R}^{N_x\times D}$, one integer M.
- compute $x \in [0,1]^{N_x\times D}$, an iid sample of the uniform law on the unit cube.
- compute $z \in [0,1]^{N_x\times D}, h \in [0,1]^{N_x}$ the polar factorization $y = \nabla_z h$, h convex, using the algorithm (1).
- compute $w_M \in [0,1]^{M\times D}$, an iid sample of the uniform law on the unit cube.
- output $(\nabla_z h)(w_M)$.

To illustrate the relevance of our algorithm, we choose $N_x = 1000$ and a bi-modal distribution, computed as follows (with the acronym iid standing for independent and identically distributed random variables):

$$x^{2n} \text{ iid } \mathcal{N}(\mu^+, I_d), \qquad x^{2n+1} \text{ iid } \mathcal{N}(\mu^-, I_d),$$

where $\mathcal{N}(\mu^\pm, I_d)$ denotes the normal law centered at $\mu^\pm = (\pm 5, \ldots, \pm 5) \in \mathbb{R}^D$ with unit variance.

In the figure, we plot the two distributions: the blue plot shows the original distribution, while red color is used for numerical output distribution. Since these are three-dimensional distributions, we plot their marginal on the combinations of axis (e_1, e_2) and (e_1, e_3).



To check the quality of our algorithm, we perform several tests. First we consider marginal distributions of x and z on each axis. To provide also information concerning the joint distributions, we compute the discrepancy error between the two distributions. Observe that no other general, statistical indicator is available that would allow one to compare two probability distributions. Skewness is a measure of the degree of symmetry, while kurtosis is a measure of the concentration of the distribution.

- Skewness and kurtosis comparison tests on each axis:

Table 5.1: skewness and kurtosis on each axis

distribution	skew0	kurtosis0	skew1	kurtosis1	skew2	kurtosis2
original	0.0028489	-1.847989	0.0024598	-1.844245	-0.0022913	-1.839244
generated	0.0096915	-1.882242	0.0191324	-1.865481	0.0172387	-1.879981

- Kolmogorov-Smirnov tests on each axis shows that we can not disprove the hypothesis that two independent samples are drawn from the same continuous distribution.
- We provide also the discrepancy error between x and z which is very small, showing that the two distributions agree very well.

Table 5.2: Kolmogorov-Smirnoff test

statistic	pvalue
0.048	0.1995737
0.034	0.6101665
0.037	0.5005674

- Finally, we plot the discrepancy error $d_k(\mu_X, \mu_Z)$ as defined in (2.32):

Table 5.3: discrepancy error

$d_k(\mu_X, \mu_Z)$
0.0812096

6 References

[1] A. Antonov and M. Konikov and M. Spector, The free boundary SABR: natural extension to negative rates, January 2015. Available at https://ssrn.com/abstract=2557046.

[2] I. Babuska, U. Banerjee, and J.E. Osborn, Survey of mesh-less and generalized finite element methods: a unified approach, *Acta Numer.* 12 (2003), 1–125.

[3] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics,* Springer Science, Business Media, LLC, 2004.

[4] M.A. Bessa, and J.T. Foster, T. Belytschko, and W.K. Liu, A mesh-free unification: reproducing kernel peridynamics, *Comput. Mech.* 53 (2014), 1251–1264.

[5] A. Brace, and D. Gatarek and M. Musiela, The market model of interest rate dynamics, *Math. Finance* 7 (1997), 127–154.

[6] H. Buehler, Volatility and dividends: volatility modeling with cash dividends and simple credit risk, February 2010, available at: https://ssrn.com/abstract=1141877.

[7] G.E. Fasshauer, *Mesh-free methods,* in "Handbook of Theoretical and Computational Nanotechnology", Vol. 2, 2006.

[8] E.G. Fasshauer, *Mesh-free approximation methods with Matlab,* Interdisciplinary Math. Sciences, Vol. 6, World Scientific Publishing Co. Pte. Ltd., Hackensack, NJ, 2007.

[9] F.C. Günther and W.K. Liu, Implementation of boundary conditions for mesh-less methods, *Comput. Methods Appl. Mech. Engrg.* 163 (1998), 205–230.

[10] E. Haghighat, M. Raissib, A. Moure, H. Gomez, and R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, *Comput. Methods Appl. Mech. Engrg.* 379 (2021), 113741

[11] T.F. Korzeniowski and K. Weinberg, A multi-level method for data-driven finite element computations, *Comput. Methods Appl. Mech. Engrg.* 379 (2021), 113740.

[12] J.J. Koester and J.-S. Chen, Conforming window functions for mesh-free methods, *Comm. Numer. Methods Engrg.* 347 (2019), 588–621.

[13] Y. LeCun, C. Cortes, and C.J.C. Burges, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/

[14] P.G. LeFloch and J.-M. Mercier, Revisiting the method of characteristics via a convex hull algorithm, *J. Comput. Phys.* 298 (2015), 95–112.

[15] P.G. LeFloch and J.-M. Mercier, A new method for solving Kolmogorov equations in mathematical finance, *C. R. Math. Acad. Sci. Paris* 355 (2017), 680–686.

[16] P.G. LeFloch and J.-M. Mercier, The Transport-based Mesh-free Method (TMM). A short review, *The Wilmott journal* 109 (2020), 52–57. Available at ArXiv:1911.00992.

[17] P.G. LeFloch and J.-M. Mercier, Mesh-free error integration in arbitrary dimensions: a numerical study of discrepancy functions, *Comput. Methods Appl. Mech. Engrg.* 369 (2020), 113245.

[18] P.G. LeFloch, J.-M. Mercier, and S. Miryusupov, CodPy: a tutorial, January 2021, available at ssrn.com/abstract=3769804.

[19] P.G. LeFloch, J.-M. Mercier, and S. Miryusupov, CodPy: an advanced tutorial, January 2021, available at ssrn.com/abstract=3769804.

[20] P.G. LeFloch, J.-M. Mercier, and S. Miryusupov, CodPy: a kernel-based reordering algorithm, January 2021, available at ssrn.com/abstract=3770557.

[21] P.G. LeFloch, J.-M. Mercier, and S. Miryusupov, Tackling the curse of dimensionality in Python: a tutorial for CodPy, in preparation.

[22] S.F. Li and W.K. Liu, *Mesh-free particle methods,* Springer Verlag, Berlin, 2004.

[23] G.R. Liu, *Mesh-free methods: moving beyond the finite element method,* CRC Press, Boca Raton, FL, 2003.

[24] G.R. Liu, An overview on mesh-free methods for computational solid mechanics, *Int. J. Comp. Methods* 13 (2016), 1630001.

[25] M. Matsumoto and T. Nishimura, Mersenne twister: a 623-dimensionally equi-distributed uniform pseudo-random number generator, *ACM Trans. Model. Comput. Simul.* 8 (1998), 3–30.

[26] J.-M. Mercier and S. Miryusupov, Hedging strategies for net interest income and economic values of equity, September 2019, available at ssrn.com/abstract=3454813.

[27] Y. Nakano, Convergence of mesh-free collocation methods for fully nonlinear parabolic equations, *Numer. Math.* 136 (2017), 703–723.

[28] F. Narcowich, J. Ward, and H. Wendland, Sobolev bounds on functions with scattered zeros, with applications to radial basis function surface fitting, *Math. of Comput.* 74 (2005), 743–763.

[29] H. Niederreiter, *Random number generation and quasi-Monte Carlo methods,* CBMS-NSF Regional Conf. Series in Applied Math., Soc. Industr. Applied Math., 1992.

[30] H.S. Oh, C. Davis, and J.W. Jeong, Mesh-free particle methods for thin plates, *Comput. Methods Appl. Mech. Engrg.* 209/212 (2012), 156–171.

[31] R. Opfer, Multiscale kernels, *Adv. Comput. Math.* 25 (2006), 357–380.

[32] R. Salehi and M. Dehghan, A moving least square reproducing polynomial mesh-less method, *Appl. Numer. Math.* 69 (2013), 34–58.

[33] J. Sirignano and K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, *J. Comput. Phys.* 375 (2018), 1339–1364.

[34] R.S. Varga, *Matrix iterative analysis,* Springer Verlag, 2000.

[35] C. Villani, *Optimal transport, old and new,* Springer Verlag, 2009.

[36] H. Wendland, Sobolev-type error estimates for interpolation by radial basis functions, in "Surface fitting and multiresolution methods" (Chamonix-Mont-Blanc, 1996), Vanderbilt Univ. Press, Nashville, TN, 1997, pp. 337–344.

[37] H. Wendland, *Scattered data approximation,* Cambridge Monograph, Applied Comput. Math., Cambridge University, 2005.

[38] J.X. Zhou and M.E. Li, Solving phase field equations using a mesh-less method, *Comm. Numer. Methods Engrg.* 22 (2006), 1109–1115.

[39] B. Zwicknagl, Power series kernels, *Constructive Approx.* 29 (2008), 61–84.