SIAM J. SCI. COMPUT.
Vol. 40, No. 5, pp. A3400–A3422

FAST ENTROPIC REGULARIZED OPTIMAL TRANSPORT USING SEMIDISCRETE COST APPROXIMATION*

EVGENY TENETOV†, GERSHON WOLANSKY†, AND RON KIMMEL‡

Abstract. The optimal transportation theory was successfully applied to different tasks on geometric domains as images and triangle meshes. In these applications the transport problem is defined on a Riemannian manifold with geodesic distance $d(x, y)$. Usually, the cost function used is the geodesic distance d or the squared geodesic distance d^2. These choices result in the 1-Wasserstein distance, also known as the earth mover's distance (EMD), or the 2-Wasserstein distance. The entropy regularized optimal transport problem can be solved using the Bregman projection algorithm. This algorithm can be implemented using only matrix multiplications of matrix $\exp(-\boldsymbol{C}/\varepsilon)$ (pointwise exponent) and pointwise vector multiplications, where $\boldsymbol{C}$ is a cost matrix, and ε is the regularization parameter. In this paper, we obtain a low-rank decomposition of this matrix and exploit it to accelerate the Bregman projection algorithm. Our low-rank decomposition is based on the semidiscrete approximation of the cost function, which is valid for a large family of cost functions, including $d^p(x, y)$, where $p \geq 1$. Our method requires the calculation of only a small portion of the distances.

Key words. optimal transport, entropy regularization, Kullback–Leibler, Bregman projection, convex optimization, Wasserstein barycenter, geodesic distance, low-rank approximations

AMS subject classifications. 65D05, 90C08, 49M25

DOI. 10.1137/17M1162925

1. Introduction.

Currently, there are three broad categories of approaches for solving numerically optimal mass transport problems. The first category is composed of discrete combinatorial algorithms. These work well for arbitrary cost functions, but they do not scale well for large problems. The second category is a class of continuous solvers. These methods use the polar factorization theorem and the Monge–Ampère equation [14, 4]. Other methods use a dynamical formulation with an additional time variable (the Benamou–Brenier formula) [5]. For the L_1 cost function fast approaches exist [21, 22, 25]. Most of the continuous solvers are restricted to a specific cost function (such as L_1 or L_2) but do not require computing and storing the cost matrix.

The third category is semidiscrete algorithms [20, 23]. In these algorithms, one of the measures is considered to be discrete. The dual problem becomes an optimization problem that can be interpreted as the problem of finding a certain power diagram partition of the continuous measure. For a specific cost function, such as the L_2 cost in $\mathbb{R}^n$, tools from computational geometry can be used to speed up calculations of the objective function.

The current paper deals with approximate optimal transport problems. The entropy regularized optimal transport problem was introduced in [3, 10, 11]. Under entropy regularization, optimal transportation is solved using the Bregman projections algorithm, which is a generalization of the Sinkhorn–Knopp algorithm (also known as the iterative proportional fitting procedure (IPFP)). Each iteration involves

*Submitted to the journal's Methods and Algorithms for Scientific Computing section December 26, 2017; accepted for publication (in revised form) August 15, 2018; published electronically October 9, 2018.

http://www.siam.org/journals/sisc/40-5/M116292.html

†Department of Mathematics, Technion, Israel Institute of Technology, Haifa 32000, Israel (teevgen@campus.technion.ac.il, gershonw@math.technion.ac.il).

‡Computer Science Department, Technion, Israel Institute of Technology, Haifa 32000, Israel (ron@cs.technion.ac.il).

a multiplication of the matrix $\exp(-\boldsymbol{C}/\varepsilon)$ (pointwise exponent) by a vector and pointwise vector multiplications. Although the complexity of each iteration is $O(n^2)$, each iteration is completely parallelizable.

The approximate methods and the exact methods are divided into two classes: very specific methods, which require a particular cost function (e.g., dynamical approaches and fast semidiscrete algorithms) and general methods which are valid for every cost function (e.g., discrete algorithms and iterative Bregman projections). The general methods do not use the geometric structure of the cost function and hence require storing a large amount of information (e.g., large matrix) and performing demanding calculations. Specific methods are usually faster and do not require storing the cost matrix but are limited to specific cost functions. Our method is intermediate; it works for a large family of cost functions and requires storing only a small portion of the cost matrix.

Recently, a fast variation of the Bregman projections algorithm for the d^2 cost function was proposed [30]; it exploits the connection of $\exp(-\boldsymbol{C}/\varepsilon)$ to the heat kernel of a surface.

In this paper, we use a different approach to accelerate the Bregman projections algorithm. Inspired by the efficiency of low-rank approximations of geodesic distance matrices [26, 27], we propose computing a low-rank approximation $\boldsymbol{R}_1\boldsymbol{R}_2^t$ of $\exp(-\boldsymbol{C}/\varepsilon)$ where $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ are $n \times m$ matrices with $m \ll n$. This approximation can be used to accelerate the Bregman projections algorithm by reducing the complexity of each iteration to $O(mn)$.

It appears that one such factorization can be obtained using the semidiscrete approximation of the cost function. This approximation expresses a geometric property of a large family of cost functions. In particular, it is valid for costs of the form d^p where d is geodesic distance and $p \geq 1$. Thus, we propose a unified approach for the 1- and 2-Wasserstein cases. Our method requires calculating only a small portion of the cost matrix.

The paper is organized as follows. In section 2 we briefly review optimal transport theory. In section 3 we describe a low-rank approximation $\boldsymbol{R}_1\boldsymbol{R}_2^t$ of $\exp(-\boldsymbol{C}/\varepsilon)$. In section 4 we describe the low-rank iterative Bregman projections algorithm. In section 5 we compare our decomposition to the Nystrom low-rank approximation. In section 6 we improve the approximation even more using a decomposition of the form $\boldsymbol{R}_1\boldsymbol{W}\boldsymbol{R}_2^t + \boldsymbol{S}$, where $\boldsymbol{S}$ is a sparse $n \times n$ matrix and $\boldsymbol{W}$ is a diagonal $m \times m$ matrix. Finally, in section 7 we support the proposed method with experimental results, which are followed by conclusions.

2. Preliminaries.

2.1. Optimal transport. We begin with background on optimal transportation. Let X and Y be measure spaces and $\mu \in P(X)$, $\nu \in P(Y)$ be probability measures on X and Y, respectively. Let $c : X \times Y \to [0, \infty)$ be a cost function. The Kantorovich problem [16, 31] is

$$W_c(\mu, \nu) := \inf_{\gamma \in \Pi(\mu,\nu)} \int_{X \times Y} c(x, y)\, d\gamma(x, y), \tag{1}$$

where $\Pi(\mu, \nu)$ is the set of the transport plans, which means the set of measures γ in $P(X \times Y)$ such that

$$\gamma(A \times Y) = \mu(A), \;\; \gamma(X \times B) = \nu(B)$$

for all measurable sets A in X and B in Y.

We fix two measures τ_1 and τ_2 on X and Y. We can write (1) as

$$\inf_{\gamma\in\Pi(\mu,\nu)}\int_{X\times Y} c(x,y)\frac{d\gamma(x,y)}{d(\tau_1\times\tau_2)}\,d(\tau_1\times\tau_2).$$

For our purposes we assume $X=Y$, $\tau:=\tau_1=\tau_2$ and denote $\bar{\tau}:=\tau\times\tau$. Next, we discuss the discrete case. We use the notion $\mathbb{R}_+$ for nonnegative real numbers and $\mathbb{R}_{++}$ for the set of strictly positive real numbers. Suppose we have a sampling $\tilde{X}=\{x_1,\dots,x_n\}$ of X. Define

$$c_{ij}=c(x_i,y_j),\quad \pi_{ij}=\frac{d\gamma}{d\bar{\tau}}(x_i,y_j),$$
$$p_i=\frac{d\mu}{d\tau}(x_i),\quad q_i=\frac{d\nu}{d\tau}(x_i),\quad a_i=\tau(x_i),\ \ 1\le i,j\le n.$$

We shall denote the finite matrices and vectors by bold letters, e.g., $\boldsymbol{C}:=\{c_{ij}\}$, $\boldsymbol{p}:=\{p_i\}$, etc. Following [30], we obtain after discretization

$$W_{\boldsymbol{C}}(\boldsymbol{p},\boldsymbol{q})=\min_{\boldsymbol{\pi}\in\Pi(\boldsymbol{p},\boldsymbol{q})}\sum_{1\le i,j\le n} a_ia_jc_{ij}\pi_{ij}, \tag{2}$$

where

$$\Pi(\boldsymbol{p},\boldsymbol{q})=\{\boldsymbol{\pi}\in\mathbb{R}_+^{n\times n}\mid \boldsymbol{a}^t\boldsymbol{\pi}\boldsymbol{a}=1,\boldsymbol{\pi}\boldsymbol{a}=\boldsymbol{p},\ \boldsymbol{\pi}^t\boldsymbol{a}=\boldsymbol{q}\}$$

and $\boldsymbol{p},\boldsymbol{q}$ satisfy

$$\boldsymbol{p}^t\boldsymbol{a}=1,\ \ \boldsymbol{q}^t\boldsymbol{a}=1.$$

For an image $X=[0,n_1]\times[0,n_2]$ discretized by a grid of $n_1\times n_2$ points, we take $n=n_1n_2$ and $\boldsymbol{a}=\frac{\mathbb{1}}{n_1n_2}$, where $\mathbb{1}=(1,\dots,1)^t\in\mathbb{R}^n$. For X a 2-dimensional manifold, discretized using a triangular mesh, we take n to be the number of vertices and the area vector $\boldsymbol{a}$ as areas proportional to the sum of triangle areas adjacent to a given vertex. Suppose γ is absolutely continuous with respect to $\bar{\tau}$. The relative entropy of a measure γ with respect to $\bar{\tau}$ is defined as

$$E(\gamma;\bar{\tau}):=-\int_{X\times X}\frac{d\gamma}{d\bar{\tau}}\ln\frac{d\gamma}{d\bar{\tau}}d\bar{\tau}. \tag{3}$$

The discretization of the relative entropy takes the form

$$E(\boldsymbol{\pi};\boldsymbol{a}\boldsymbol{a}^t):=-\sum_{1\le i,j\le n} a_ia_j\pi_{ij}\ln\pi_{ij}.$$

Following [3, 10], we modify the objective of the optimal transportation problem in (1) by subtracting an ε multiple of the entropy term

$$W_{c,\varepsilon}(\mu,\nu):=\inf_{\gamma\in\Pi(\mu,\nu)}\int_{X\times X}c(x,y)\,d\gamma-\varepsilon E(\gamma;\bar{\tau}),$$

so the discrete regularized optimal transport distance can be written as

$$W_{\boldsymbol{C},\varepsilon}(\boldsymbol{p},\boldsymbol{q}):=\min_{\boldsymbol{\pi}\in\Pi(\boldsymbol{p},\boldsymbol{q})}\sum_{1\le i,j\le n}a_ia_jc_{ij}\pi_{ij}+\varepsilon\sum_{1\le i,j\le n}a_ia_j\pi_{ij}\ln\pi_{ij}. \tag{4}$$

We refer the reader to [19] for a discussion of the connection of (4) to the nonregularized transport as $\varepsilon\to0$.

Remark. The definition of a_i depends on the quantization of the measures. If the quantization is consistent with the centroids of the Voronoi cells, then we choose $a_i=1/n$.

2.2. Bregman projections. Let γ, ξ be measures on $X \times X$, where γ is absolutely continuous with respect to ξ. We define the KL (Kullback–Leibler) divergence between γ and ξ as

$$\mathrm{KL}(\gamma|\xi) := \int_{X\times X} \frac{d\gamma}{d\xi}\left(\ln \frac{d\gamma}{d\xi} - 1\right) d\xi + 1.$$

We note that for $\xi := \exp(-c(x,y)/\varepsilon) d\overline{\tau}$ we obtain

$$W_{c,\varepsilon}(\mu,\nu) = \varepsilon \min_{\gamma\in\Pi(\mu,\nu)} \mathrm{KL}(\gamma|\xi).$$

Now we define the discrete KL divergence between $\boldsymbol{\pi} \in \mathbb{R}_+^{n\times n}$ and $\boldsymbol{\xi} \in \mathbb{R}_{++}^{n\times n}$:

$$\mathrm{KL} : \mathbb{R}_+^{n\times n} \times \mathbb{R}_{++}^{n\times n} :\to \mathbb{R}, \quad \mathrm{KL}(\boldsymbol{\pi}|\boldsymbol{\xi}) := \sum_{1\le i,j\le n} a_i a_j \left(\pi_{ij} \ln\left(\frac{\pi_{ij}}{\xi_{ij}}\right) - \pi_{ij}\right) + 1.$$

For a cost matrix $\boldsymbol{C}$ and $\varepsilon > 0$ we denote

$$\boldsymbol{H}(\boldsymbol{C},\varepsilon) := \{\exp(-c_{ij}/\varepsilon)\}, \tag{5}$$

where the exponential is pointwise. Define

$$\mathcal{C}_1 = \{\boldsymbol{\pi} \in \mathbb{R}^{n\times n} \mid \boldsymbol{\pi}\boldsymbol{a} = \boldsymbol{p}\} \text{ and } \mathcal{C}_2 = \{\boldsymbol{\pi} \in \mathbb{R}^{n\times n} \mid \boldsymbol{\pi}^t\boldsymbol{a} = \boldsymbol{q}\}.$$

Problem (4) can be written as

$$W_{\boldsymbol{C},\varepsilon}(\boldsymbol{p},\boldsymbol{q}) = \varepsilon\left(\min_{\boldsymbol{\pi}\in\mathcal{C}_1\cap\mathcal{C}_2} \mathrm{KL}(\boldsymbol{\pi}|H(\boldsymbol{C},\varepsilon))\right). \tag{6}$$

Note that nonnegativity constraints are already in the definition of the entropy. Given a convex set $\mathcal{C} \subset \mathbb{R}^{n\times n}$ the projection according to the KL divergence is defined as

$$P_{\mathcal{C}}^{\mathrm{KL}}(\boldsymbol{\xi}) := \mathrm{argmin}_{\boldsymbol{\pi}\in\mathcal{C}} \mathrm{KL}(\boldsymbol{\pi}|\boldsymbol{\xi}).$$

Let us consider the special case $\mathcal{C} = \mathcal{C}_1 \cap \mathcal{C}_2$. It is possible to solve for $\boldsymbol{\pi} \in \mathcal{C}$ satisfying (6) by simply using iterative KL projections; see [6]. Starting from $\boldsymbol{\pi}_0 = \exp(-\boldsymbol{C}/\varepsilon)$, one computes the alternating projections

$$\boldsymbol{\pi}_N = P_{\mathcal{C}_N}^{\mathrm{KL}}(\boldsymbol{\pi}_{N-1}) \quad \forall N > 0,$$

where $\mathcal{C}_N = \mathcal{C}_1$ if n is odd, and $\mathcal{C}_N = \mathcal{C}_2$ otherwise. One can then show that $\boldsymbol{\pi}_N$ converges towards the unique solution of (6),

$$\boldsymbol{\pi}_N \to \boldsymbol{\pi} = P_{\mathcal{C}}^{\mathrm{KL}}(\boldsymbol{\pi}) \quad \text{as } N \to \infty.$$

If the original problem has a unique solution, then the sequence π_N converges to it; otherwise it converges to the solution with largest entropy.

In this special case of two sets, the algorithm is known as an iterative proportional fitting procedure (IPFP) or Sinkhorn's algorithm; see [28].

For two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ we denote by $\boldsymbol{v} \odot \boldsymbol{w}$ the pointwise multiplication and by $\boldsymbol{v} \oslash \boldsymbol{w}$ the pointwise division. $\mathrm{diag}(\boldsymbol{v})$ denotes the diagonal matrix with elements in the vector $\boldsymbol{v}$. The following proposition is a variant of Proposition 1 in [3].

Algorithm 1 Optimal transport distance using Bregman projections.

function OPTIMAL-TRANSPORT-DISTANCE($\boldsymbol{p}, \boldsymbol{q}; \boldsymbol{H}(\boldsymbol{C}, \varepsilon)$)
  $\boldsymbol{v}, \boldsymbol{w} \leftarrow \mathbb{1}$
  for $i = 1, 2, 3, \ldots$ **do**
    $\boldsymbol{v} \leftarrow \boldsymbol{p} \oslash (\boldsymbol{H}(\boldsymbol{C}, \varepsilon)(\boldsymbol{a} \odot \boldsymbol{w}))$
    $\boldsymbol{w} \leftarrow \boldsymbol{q} \oslash (\boldsymbol{H}(\boldsymbol{C}, \varepsilon)^t(\boldsymbol{a} \odot \boldsymbol{v}))$
  return $\varepsilon \boldsymbol{a}^t \left[(\boldsymbol{p} \odot \ln(\boldsymbol{v})) + (\boldsymbol{q} \odot \ln(\boldsymbol{w}))\right]$

PROPOSITION 1 (Proposition 1 in [30]). *For* $\boldsymbol{\pi} \in \mathbb{R}_+^{n \times n}$

$$P_{\mathcal{C}_1}^{\mathrm{KL}}(\boldsymbol{\pi}) = \operatorname{diag}(\boldsymbol{p} \oslash (\boldsymbol{\pi}\boldsymbol{a}))\boldsymbol{\pi} \quad \textit{and} \quad P_{\mathcal{C}_2}^{\mathrm{KL}}(\boldsymbol{\pi}) = \boldsymbol{\pi} \operatorname{diag}(\boldsymbol{q} \oslash (\boldsymbol{\pi}^t \boldsymbol{a})).$$

Proposition 1 allows us to apply Algorithm 1.

An important advantage of this algorithm is that it uses only matrix-vector multiplications applied to a fixed matrix $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$ and pointwise multiplications, which can all be easily parallelized on modern hardware. Next, we will use a low-rank factorization of $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$ to further accelerate the matrix multiplications in Algorithm 1.

2.3. Wasserstein barycenter. As a further application, we consider the regularized Wasserstein barycenter problem [2, 17].

Given a set of weights $(\alpha_1, \ldots, \alpha_k) \in \mathbb{R}_+^k$ it is defined as the following convex problem over the space of measures:

$$\min_{\mu} \sum_{i=1}^{k} \alpha_i W_c(\mu, \mu_i).$$

After discretization and substituting the regularized Wasserstein distance we obtain

$$\min_{\boldsymbol{p}} \sum_{i=1}^{k} \alpha_i W_{\boldsymbol{C},\varepsilon}(\boldsymbol{p}, \boldsymbol{p}_i),$$

which, as in [3] and [30], can be rewritten as

$$\boldsymbol{p} = \boldsymbol{\pi}_i \boldsymbol{a}, \quad 1 \le i \le k,$$

where the set of optimal couplings $\boldsymbol{\pi} = (\boldsymbol{\pi}_i)_{i=1}^k$ solves

$$\min \left\{ \sum_{i=1}^{k} \alpha_i \operatorname{KL}(\boldsymbol{\pi}_i \mid \boldsymbol{H}(\boldsymbol{C}, \varepsilon)) \mid \boldsymbol{\pi} \in \mathcal{C}_1 \cap \mathcal{C}_2 \right\}.$$

We denote

$$\mathcal{C}_1 = \{\{\boldsymbol{\pi}_i\}_{i=1}^k \mid \boldsymbol{\pi}_i^t \boldsymbol{a} = \boldsymbol{p}_i \;\; 1 \le i \le k\},$$
$$\mathcal{C}_2 = \{\{\boldsymbol{\pi}_i\}_{i=1}^k \mid \boldsymbol{\pi}_i \boldsymbol{a} = \boldsymbol{\pi}_j \boldsymbol{a}, \;\; 1 \le i, j \le k\}.$$

The KL projections on $\mathcal{C}_1$ and $\mathcal{C}_2$ can be obtained in a closed form. The following propositions are variants of Proposition 2 in [3].

PROPOSITION 2 (Proposition 2 in [30]). *The* KL *projection of* $(\boldsymbol{\pi}_i)_{i=1}^k$ *onto* $\mathcal{C}_1$ *satisfies*

$$P_{\mathcal{C}_1}^{\mathrm{KL}}(\boldsymbol{\pi}_i) = \boldsymbol{\pi}_i \operatorname{diag}(\boldsymbol{p}_i \oslash \boldsymbol{\pi}_i^t \boldsymbol{a}), \;\; 1 \le i \le k.$$

PROPOSITION 3 (Proposition 3 in [30]). *The* KL *projection of* $(\boldsymbol{\pi}_i)_{i=1}^k$ *onto* $\mathcal{C}_2$ *satisfies*

$$P_{\mathcal{C}_2}^{\mathrm{KL}}(\boldsymbol{\pi}_i) = \mathrm{diag}(\boldsymbol{\mu} \oslash \boldsymbol{d}_i)\boldsymbol{\pi}_i, \quad 1 \le i \le k,$$

where $\boldsymbol{d}_i = \boldsymbol{\pi}_i \boldsymbol{a}$ *and* $\boldsymbol{\mu} = \Pi_i \boldsymbol{d}_i^{\alpha_i / \sum_l \alpha_l}$.

Algorithm 2 Wasserstein barycenter using Bregman projections.

function WASSERSTEIN-BARYCENTER($\{\boldsymbol{p}_i\}_{i=1}^k, \{\alpha_i\}_{i=1}^k; \boldsymbol{H}(\boldsymbol{C}, \varepsilon), \boldsymbol{a}$)
  $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k \leftarrow \mathbb{1}$
  $\boldsymbol{w}_1, \dots, \boldsymbol{w}_k \leftarrow \mathbb{1}$
  for $j = 1, 2, 3, \dots$ **do**
    $\boldsymbol{p} \leftarrow \mathbb{1}$
    for $i = 1, \dots, k$ **do**
      $\boldsymbol{w}_i \leftarrow \boldsymbol{p}_i \oslash (\boldsymbol{H}(\boldsymbol{C}, \varepsilon)^t (\boldsymbol{a} \odot \boldsymbol{v}_i))$
      $\boldsymbol{d}_i \leftarrow \boldsymbol{v}_i \odot (\boldsymbol{H}(\boldsymbol{C}, \varepsilon)(\boldsymbol{a} \odot \boldsymbol{w}_i))$
      $\boldsymbol{p} \leftarrow \boldsymbol{p} \odot \boldsymbol{d}_i^{\alpha_i}$
    for $i = 1, \dots, k$ **do**
      $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i \odot \boldsymbol{p} \oslash \boldsymbol{d}_i$
  return $\boldsymbol{p}$

3. Low-rank decomposition of $\boldsymbol{H}(\boldsymbol{C}, \varepsilon) := \exp(-\boldsymbol{C}/\varepsilon)$.

3.1. Semidiscrete cost approximation. Let X and Y be measure spaces. We consider the cost functions of the form

$$c : X \times Y \to [0, \infty), \quad c(x, y) = \min_{z \in Z} (c^{(1)}(x, z) + c^{(2)}(y, z)), \tag{7}$$

where Z is another space, and $c^{(1)} : X \times Z \to [0, \infty)$, $c^{(2)} : Y \times Z \to [0, \infty)$ are two other functions. A canonical example is $X = Y = Z = \mathbb{R}^d$ and $c(x, y) = \|x - y\|^p$ with $p \ge 1$. Then we have

$$\|x - y\|^p = 2^{p-1} \min_{z \in \mathbb{R}^d} (\|x - z\|^p + \|z - y\|^p). \tag{8}$$

Another example is

$$\|x - y\|_1 = \min_{z \in \mathbb{R}^d} (\|x - z\|_1 + \|z - y\|_1). \tag{9}$$

Let X be a compact Riemannian manifold with geodesic distance $d(x, y)$. Let $h : [0, \infty) \to [0, \infty)$ be a convex and strictly increasing function. We consider the cost function $c(x, y) = h(d(x, y))$ on $X \times X$. Then, it can be proved [32] that

$$h(d(x, y)) = \min \left\{ \int_0^1 h(\|\dot{\gamma}(t)\|) dt; \ \ \gamma(0) = x, \ \ \gamma(1) = y, \ \ \gamma \in C^1[0, 1] \right\}.$$

It follows that

$$h(d(x, y)) = \min_{z \in X} \left(\frac{1}{2} h(2d(x, z)) + \frac{1}{2} h(2d(y, z)) \right). \tag{10}$$

Note that in these cases,

$$X = Y = Z \ \text{ and } \ c^{(1)}(x, z) = c^{(2)}(x, z) = \frac{1}{2} h(2d(x, z)). \tag{11}$$

A particular example is $c(x,y) = d(x,y)^p$ for $p \geq 1$. Then, we have

$$d(x,y)^p = 2^{p-1} \min_{z \in X} (d^p(x,z) + d^p(y,z)). \tag{12}$$

In this paper, we assume X, Y, and Z are compact. Given a finite sampling $Z_m \subset Z$ with $m \ll n$ elements, we can approximate $c(x,y)$ using

$$c^{Z_m}(x,y) := \min_{z \in Z_m} (c^{(1)}(x,z) + c^{(2)}(y,z)). \tag{13}$$

Now we discuss the discrete setting. We denote the samplings of the spaces X, Y, and Z by

$$X_n = \{x_1, \ldots, x_n\}, \;\; Y_n = \{y_1, \ldots, y_n\}, \;\; Z_m = \{z_1, \ldots, z_m\}.$$

Let $\boldsymbol{C}^{(1)}$ and $\boldsymbol{C}^{(2)}$ be the $n \times m$ matrices related to $c^{(1)} : X \times Z \to [0, \infty)$ and $c^{(2)} : X \times Z \to [0, \infty)$, respectively. That means

$$c^{(1)}_{ij} := c^{(1)}(x_i, z_j), \;\; c^{(2)}_{ij} := c^{(2)}(y_i, z_j), \;\; 1 \leq i \leq n, \;\; 1 \leq j \leq m.$$

Then, the semidiscrete approximation is

$$c^{Z_m}_{ij} := \min_{1 \leq k \leq m} (c^{(1)}_{ik} + c^{(2)}_{jk}), \;\; 1 \leq i, j \leq n. \tag{14}$$

We now limit ourselves to the case (10) of a compact Riemannian manifold X with a cost function $c(x,y) = h(d(x,y))$. In the experiments we focus on the case $h(x) = |x|^p$ where $p \geq 1$, in which we obtain the approximation (12). We denote

$$h_1(x) := \frac{1}{2} h(2x) \text{ and } c^{(1)}(x,z) = c^{(2)}(x,z) = h_1(d(x,z)), \;\; x, z \in X. \tag{15}$$

Then

$$c^{Z_m}_{ij} = \min_{1 \leq k \leq m} (h_1(d_{ik}) + h_1(d_{jk})), \quad 1 \leq i, j \leq n, \tag{16}$$

where

$$d_{ij} := d(x_i, z_j), \;\; 1 \leq i \leq n, \; 1 \leq j \leq m. \tag{17}$$

The approximation (16) requires one to calculate the mn distances (17).

3.2. Construction of the sampling set. The computation of geodesic distances from a point to the rest can be performed efficiently using the fast marching method [18] for 2-dimensional triangulated surfaces and Dijkstra's shortest path algorithm [12] for higher dimensions. For 2-dimensional surfaces with n vertices, both algorithms have complexities of $O(n \log n)$. For given sampling X_n of X, the algorithm below produces a sampling $Z_m \subset X_n$ and calculates the necessary distances. We use the farthest point sampling strategy which is a 2-optimal method [15].

Algorithm 3 Distance calculation.

function DISTANCE-CALCULATION(X_n, m)
  // Farthest point sampling
  Choose an initial vertex $z_1 \in X_n$ and define $Z_m := \{z_1\}$
  $d_{i1} \leftarrow d(x_i, z_1)$, $1 \le i \le n$,
  for $j = 2, \dots, m$ **do**
    $z_j \leftarrow \operatorname{argmax}\{x \in X_n \mid \min_{1 \le l \le i-1} d(x, z_l)\}$
    $\lambda \leftarrow \max\{x \in X_n \mid \min_{1 \le l \le i-1} d(x, z_l)\}$
    $d_{ij} \leftarrow d(x_i, z_j)$, $1 \le i \le n$,
    $Z_m \leftarrow Z_m \cup \{z_j\}$
  return Z_m, $\boldsymbol{D}_{n \times m} := \{d_{ij}\}$, λ

Remarks.

1. The algorithm stops after *at most* n steps and $Z_m \subset X_n$ If we use instead of X_n some other sampling of X, then the set Z_m obtained is *not necessarily* contained in X_n.
2. $\lambda(m)$ is monotone decreasing as a function of m. It cannot be too small since we require $m \ll n$. A rough estimate for λ is

$$\lambda(m) \approx C \left(\frac{\mathrm{Vol}(X)}{m} \right)^{1/\dim(X)} \tag{18}$$

for some suitable constant C.

3.3. Construction of a low-rank decomposition. We utilize the following approximation for a minimum: Given a vector $\boldsymbol{a} = (a_1, \dots, a_m) \in \mathbb{R}^m$ and $\varepsilon > 0$, we define

$$\min_{\varepsilon}(\boldsymbol{a}) := -\varepsilon \ln \left(\sum_{i=1}^{m} \exp\left(-a_i/\varepsilon\right) \right). \tag{19}$$

This approximation is discussed, for example, in [33, 8] and is related to the Log-Sum-Exp (LSE) function. It can be proven that $\min_{\varepsilon}(\boldsymbol{a})$ converges to $\min(\boldsymbol{a})$ as $\varepsilon \to 0^+$. Moreover,

$$\min(\boldsymbol{a}) - \varepsilon \ln m \le \min_{\varepsilon}(\boldsymbol{a}) \le \min(\boldsymbol{a}). \tag{20}$$

We define the approximated version of (14),

$$c_{ij}^{Z_m,\varepsilon} := -\varepsilon \ln \left(\sum_{k=1}^{m} \exp\left(-\frac{c_{ik}^{(1)} + c_{jk}^{(2)}}{\varepsilon} \right) \right), \qquad 1 \le i, j \le n, \tag{21}$$

where $c^{(1)}$ and $c^{(2)}$ are as defined in (15).

Define the $n \times m$ matrices

$$\boldsymbol{R}_1 := \{\exp(-c_{ij}^{(1)}/\varepsilon)\}_{n \times m}; \quad \boldsymbol{R}_2 := \{\exp(-c_{ij}^{(2)}/\varepsilon)\}_{n \times m}.$$

The approximation (21) can be written as

$$\boldsymbol{H}(\boldsymbol{C}^{Z_m,\varepsilon}, \varepsilon) = \boldsymbol{R}_1 \boldsymbol{R}_2^t \tag{22}$$

and can serve as a low-rank decomposition of $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$. In the case (10) we have

$$\boldsymbol{R}_1 = \boldsymbol{R}_2 = \{\exp(-h_1(d_{ij})/\varepsilon)\}_{n\times m}.$$

We note that the ε from problem (4) is the same ε that we use in the approximation (21) and in the decomposition (22).

3.4. Approximation bounds. In this section we bound $c_{ij}^{Z_m,\varepsilon}$. Clearly, we have

$$c(x,y) \le c^{Z_m}(x,y), \quad x,y \in X. \tag{23}$$

PROPOSITION 4. *Let $c(x,y)$ be a cost function satisfying* (10). *Then for every $\varepsilon > 0$ and $1 \le m \le n$ we have*

$$c_{ij} - \varepsilon \ln m \le c_{ij}^{Z_m,\varepsilon} \le c_{ij} + A\frac{\mathrm{Vol}(X)^{1/\dim(X)}}{m^{1/\dim(X)}}, \qquad 1 \le i,j \le n. \tag{24}$$

Proof. We can estimate the difference between $c^{Z_m}(x,y)$ and $c(x,y)$; assuming (10) we get

$$c(x,y) - c^{Z_m}(x,y) \ge c^{(1)}(x,z_*) - c^{(1)}(x,z_j) + c^{(2)}(y,z_*) - c^{(2)}(y,z_j) \tag{25}$$

for any $z_j \in Z_m$, where z_* is a minimizer in (7). Now, by convexity of h_1

$$c^{(1)}(x,z_*) - c^{(1)}(x,z_j) \ge h_1'(d(x,z_j))\left(d(x,z_*) - d(x,z_j)\right).$$

Since $d(x,z_*) - d(x,z_j) \ge -d(z_*,z_j)$ and $h_1' > 0$ by monotonicity, we get

$$c^{(1)}(x,z_*) - c^{(1)}(x,z_j) \ge -d(z_*,z_j)h_1'(d(x,z_j)). \tag{26}$$

In the same way we get

$$c^{(2)}(y,z_*) - c^{(2)}(y,z_j) \ge -d(z_*,z_j)h_1'(d(y,z_j)). \tag{27}$$

On the other hand, $d(z_*,z_j) \le \lambda$ for some $z_j \in Z_m$ by the above algorithm, and we get from (25), (26), (27) that

$$c(x,y) - c^{Z_m}(x,y) \ge -\lambda \min_{z\in Z_m}\left[h_1'(d(x,z)) + h_1'(d(y,z))\right],$$

where

$$\lambda = \max\left\{\min_{1\le j\le m} d(x,z_j) \mid x \in X\right\}.$$

Using (18) we finally obtain

$$c(x,y) - c^{Z_m}(x,y) \ge -A\frac{\mathrm{Vol}(X)^{1/\dim(X)}}{m^{1/\dim(X)}} \tag{28}$$

for some universal constant A.

From (14) and (20)

$$c_{ij}^{Z_m} - \varepsilon \ln m \le c_{ij}^{Z_m,\varepsilon} \le c_{ij}^{Z_m}, \qquad 1 \le i,j \le n, \tag{29}$$

and from (23) and (28) we get (24). ∎

Comparing the approximation on both sides of (24), we obtain a reasonable estimate of ε in terms of m:

$$\varepsilon \approx A\frac{\mathrm{Vol}(X)^{1/\dim(X)}}{m^{1/\dim(X)}\ln m}. \tag{30}$$

For such ε we have

$$|c_{ij}^{Z_m,\varepsilon} - c_{ij}| \le A\frac{\mathrm{Vol}(X)^{1/\dim(X)}}{m^{1/\dim(X)}\ln m}, \qquad 1 \le i,j \le n. \tag{31}$$

4. Iterative low-rank Bregman projections algorithm. In the previous chapter, we presented a low-rank decomposition of $\boldsymbol{H}(\boldsymbol{C},\varepsilon)$. Next, we exploit this decomposition to introduce a fast Bregman projections algorithm. We describe the algorithm in a more general setting. Suppose we have an approximation

$$\boldsymbol{H}(\boldsymbol{C},\varepsilon) \approx \boldsymbol{R}_1\boldsymbol{W}\boldsymbol{R}_2^t + \boldsymbol{S}, \tag{32}$$

where $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ are $n \times m$ matrices, $\boldsymbol{W}$ is an $m \times m$ matrix, and $\boldsymbol{S}$ is a sparse $n \times n$ matrix. For the regularized semidiscrete approximation (22), $\boldsymbol{S} = \boldsymbol{0}$, and $\boldsymbol{W} = \boldsymbol{I_m}$ is the identity matrix. We introduce the following algorithms.

Algorithm 4 Optimal transport distance using low-rank approximation (32).

function LOWRANK-OPTIMAL-TRANSPORT-DISTANCE($\boldsymbol{p},\boldsymbol{q};\boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{W},\boldsymbol{S},\boldsymbol{a}$)
  $\boldsymbol{v},\boldsymbol{w} \leftarrow \mathbb{1}$
  for $i = 1,2,3,\ldots$ **do**
    $\boldsymbol{v} \leftarrow \boldsymbol{p} \oslash (\boldsymbol{R}_1(\boldsymbol{W}(\boldsymbol{R}_2^t(\boldsymbol{a}\odot\boldsymbol{w}))) + \boldsymbol{S}(\boldsymbol{a}\odot\boldsymbol{w}))$
    $\boldsymbol{w} \leftarrow \boldsymbol{q} \oslash (\boldsymbol{R}_2(\boldsymbol{W}(\boldsymbol{R}_1^t(\boldsymbol{a}\odot\boldsymbol{v}))) + \boldsymbol{S}(\boldsymbol{a}\odot\boldsymbol{v}))$
  return $\varepsilon\boldsymbol{a}^t\left[(\boldsymbol{p}\odot\ln(\boldsymbol{v})) + (\boldsymbol{q}\odot\ln(\boldsymbol{w}))\right]$

Algorithm 5 Wasserstein barycenter using low-rank approximation (32).

function LOWRANK-WASSERSTEIN-BARYCENTER($\{\boldsymbol{p}_i\}_{i=1}^k, \{\alpha_i\}_{i=1}^k; \boldsymbol{R}_1,\boldsymbol{R}_2,\boldsymbol{W},\boldsymbol{S},\boldsymbol{a}$)
  $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_k \leftarrow \mathbb{1}$
  $\boldsymbol{w}_1,\ldots,\boldsymbol{w}_k \leftarrow \mathbb{1}$
  for $j = 1,2,3,\ldots$ **do**
    $\boldsymbol{p} \leftarrow \mathbb{1}$
    for $i = 1,\ldots,k$ **do**
      $\boldsymbol{w}_i \leftarrow \boldsymbol{p}_i \oslash (\boldsymbol{R}_1(\boldsymbol{W}(\boldsymbol{R}_2^t(\boldsymbol{a}\odot\boldsymbol{v}_i))) + \boldsymbol{S}(\boldsymbol{a}\odot\boldsymbol{v}_i))$
      $\boldsymbol{d}_i \leftarrow \boldsymbol{v}_i \odot (\boldsymbol{R}_1(\boldsymbol{W}(\boldsymbol{R}_2^t(\boldsymbol{a}\odot\boldsymbol{w}_i))) + \boldsymbol{S}(\boldsymbol{a}\odot\boldsymbol{w}_i))$
      $\boldsymbol{p} \leftarrow \boldsymbol{p} \odot \boldsymbol{d}_i^{\alpha_i}$
    for $i = 1,\ldots k$ **do**
      $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i \odot \boldsymbol{p} \oslash \boldsymbol{d}_i$
  return $\boldsymbol{p}$

4.1. Complexity analysis of the algorithm. Each iteration consists of a finite number of matrix multiplications. Each multiplication with $\boldsymbol{R}_1$ and $\boldsymbol{R}_2^t$ takes $O(mn)$, the inner multiplication takes $O(m^2)$, and the sparse matrix multiplication takes $O(s)$, where s is the number of nonzero elements in $\boldsymbol{S}$. For a suitable choice of $\boldsymbol{S}$, we can assume that $O(s) = O(mn)$. For $m \ll n$ we obtain a complexity of $O(mn)$ instead of $O(n^2)$ of the original algorithm.

5. Comparison with the Nystrom low-rank approximation. The Nystrom method is an efficient technique to generate low-rank matrix approximations of positive-definite matrices. The method was used successfully to approximate symmetric matrices, such as $\boldsymbol{C} = \{d_{ij}^2\}$ distances on a manifold [26]. Given an $n \times n$ matrix $\boldsymbol{K}$, the method requires one to sample a subset of m columns from $\boldsymbol{K}$ and to compute the corresponding submatrix $\boldsymbol{R} \in \mathbb{R}^{n\times m}$. We denote by $\boldsymbol{W} \in \mathbb{R}^{m\times m}$ the symmetric submatrix of $\boldsymbol{R}$ which consists of the corresponding m columns and m rows of $\boldsymbol{K}$. The low-rank approximation is then obtained as

$$\boldsymbol{K}^{Nys} := \boldsymbol{R}\boldsymbol{W}^{+}\boldsymbol{R}^{t}, \tag{33}$$

where $\boldsymbol{W}^{+}$ is its Moore–Penrose pseudoinverse. See [13].

In [26], the authors introduced an extension of the Nystrom method, which we describe now. Suppose $\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^t$ is the thin eigenvalue decomposition of $\boldsymbol{W}$ with $1 \le n_1 \le m$ eigenvalues. Then, we approximate the pseudoinverse by $\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^t$ and obtain the decomposition

$$\widetilde{\boldsymbol{K}}^{Nys} := \boldsymbol{R}\boldsymbol{V}\boldsymbol{\Lambda}^{-1}\boldsymbol{V}^t\boldsymbol{R}^t = (\boldsymbol{R}\boldsymbol{V})\boldsymbol{\Lambda}^{-1}(\boldsymbol{R}\boldsymbol{V})^t. \tag{34}$$

Experiments approve that this approximation gives better results (in the sense of average relative error) than (33) for $n_1 = [\frac{m}{2}]$. This method can be used for the symmetric matrix $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$ for ε not too small.

For small ε the matrix $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$ is badly scaled, and the quality of approximation for the Nystrom decomposition degrades due to numerical errors. See section A.1 for numerical comparison.

6. Optimized semidiscrete low-rank approximation with thresholding. In this section, we further improve the low-rank approximation $c^{Z_m,\varepsilon}$ for the case (16). One limitation of the semidiscrete approximation is the inaccuracy for small d_{ij} values (see Figure 9 in the Appendix). We propose overcoming this problem by explicitly calculating these distances. Additionally, we propose improving the approximation (21) further by using a weighted sum, obtaining

$$\tilde{c}_{ij}(\boldsymbol{q}) := \begin{cases} -\varepsilon \ln\left(\sum_{k=1}^{m} q_k \exp\left(-\frac{h_1(d_{ik})+h_1(d_{jk})}{\varepsilon}\right)\right), & d_{ij} \ge \lambda_0, \\ h(d_{ij}), & d_{ij} < \lambda_0, \end{cases} \tag{35}$$

where $\boldsymbol{q} \in \mathbb{R}^m$ and $\lambda_0 > 0$ is fixed. This approximation requires the computation of distances d_{ij} such that $d_{ij} < \lambda_0$. This can be done using a modified fast marching method that terminates when the distance exceeds the threshold λ_0. Denote by $\hat{\boldsymbol{D}}_{n\times n}$ the sparse matrix that holds these distances. The approximation (35) can be written as

$$\boldsymbol{H}(\tilde{\boldsymbol{C}}, \varepsilon) = \boldsymbol{R}\,\mathrm{diag}(\boldsymbol{q})\boldsymbol{R}^t + \boldsymbol{S}, \tag{36}$$

where $\boldsymbol{S}$ is an $n \times n$ sparse matrix defined by

$$s_{ij} = \begin{cases} \exp(-h(d_{ij})/\varepsilon) - \sum_{k=1}^{m} q_k \exp\left(-\frac{h_1(d_{ik})+h_1(d_{jk})}{\varepsilon}\right), & d_{ij} < \lambda_0, \\ 0 & \text{otherwise.} \end{cases} \tag{37}$$

Denote

$$s_{\lambda_0} := |\{d_{ij} \mid d_{ij} < \lambda_0\}|.$$

This is the sparsity of the matrix $\boldsymbol{S}$. The parameter λ_0 needs to be chosen such that $s_{\lambda_0} \le mn$ in order to keep the complexity $O(mn)$ of each iteration in Algorithms 4 and 5.

6.1. Practical computation of ε, λ_0, and q. The decomposition (36) has three parameters ε, $\boldsymbol{q}$, and λ_0. In this section we discuss how we practically compute them. ε should be chosen as not too small and a good value; suggested in [3] and [30] is

$$\varepsilon = \varepsilon_1 \operatorname{median}(\boldsymbol{C}),$$

where $\varepsilon_1 = 0.01$. In our case, for $c(x,y) = h(d(x,y))$ this value can be estimated from the submatrix $\boldsymbol{D}_{m\times m}$. The choice of λ_0 depends on the desired sparsity s_{λ_0} of the matrix $\boldsymbol{S}$. Suppose we want $s_{\lambda_0} = c_0 mn$ where $0 \le c_0 \le 1$. A practical approach is to sort the elements of the submatrix $\{d_{ij}\}_{m\times m}$ ($O(m^2 \log m^2)$ complexity) and to choose $\lambda_0 = h(a)$ where a is the $\lfloor p_r m^2 \rfloor$th element, and $p_r = c_0 \frac{m}{n}$ (so that $c_0 mn = n^2 p_r$). Now we discuss the optimization process of the coefficients $\boldsymbol{q} = (q_1, \ldots, q_m)$. Ideally, we want $\boldsymbol{q}$ to be optimal in the sense of

$$\boldsymbol{q} = \operatorname{argmin}_{q\in\mathbb{R}^m} \|(c_{ij} - \tilde{c}_{ij}(\boldsymbol{q}))_{1\le i,j\le n}\|_2^2.$$

In practice, we want to use the distance values which we know. Given the values $\boldsymbol{D}_{n\times n_1}$, for some $m \le n_1 \le n$, we solve

$$\boldsymbol{q} = \operatorname{argmin}_{q\in\mathbb{R}^m} \|(c_{ij} - \tilde{c}_{ij}(\boldsymbol{q}))_{1\le i\le n,\ 1\le j\le n_1}\|_2^2. \tag{38}$$

First, we note that for $\boldsymbol{q} = \alpha\mathbb{1}$ this problem takes the form

$$\alpha = \operatorname{argmin}_{\alpha\in\mathbb{R}} \|(c_{ij} - \tilde{c}_{ij}(\mathbb{1}) - \varepsilon \ln \alpha)_{1\le i\le n,\ 1\le j\le n1}\|_2^2. \tag{39}$$

The solution of (39) is given by

$$\alpha = \exp\left(\frac{1}{\varepsilon n_1 n} \sum_{1\le i\le n,\ 1\le j\le n_1} (c_{ij} - \tilde{c}_{ij}(\mathbb{1}))\right). \tag{40}$$

Then we can solve (38) using optimization with $\boldsymbol{q}_0 = \alpha\mathbb{1}$ as an initial value. We use $n_1 = 2m$. This requires the calculation of additional distances and can be done using Algorithm 3 with n_1 instead of m and using as Z_m the first m points of Z_{n_1}. Another approach is to use $\boldsymbol{q} = \alpha\mathbb{1}$ with

$$\alpha = \exp\left(\frac{1}{\varepsilon mn} \sum_{1\le i\le n,\ 1\le j\le m} (c_{ij} - \tilde{c}_{ij}(\mathbb{1}))\right). \tag{41}$$

This gives the decomposition

$$\boldsymbol{H}(\tilde{\boldsymbol{C}}, \varepsilon) = \alpha \boldsymbol{R}\boldsymbol{R}^t + \boldsymbol{S}. \tag{42}$$

As we will see in section A.1 (Figure 9), the improvement using an optimized value of $\boldsymbol{q}$ is not significant. Hence it is sufficient to use (41). This decomposition can be calculated using the three parameters ε_1, m, and c_0. Algorithm 6 summarizes the calculation of the decomposition.

6.2. Complexity analysis of the algorithm. For 2-dimensional manifolds, the total complexity of computing the decomposition is $O(nm \log n)$. The calculation of $\boldsymbol{D}_{n\times m}$ (using fast marching) takes $O(nm \log n)$. The computation of ε and λ_0 takes $O(m \log m)$. Since the sparsity of $\boldsymbol{S}$ is $c_0 mn$ for $0 \le c_0 \le 1$, each set

$$P_i := \{d_{ij} \mid d_{ij} < \lambda_0,\ 1 \le i, j \le n\}$$

has approximately m elements; hence the calculation of $\hat{\boldsymbol{D}}_{n\times n}$ (using fast marching) takes approximately $O(nm \log m)$, and the total decomposition takes $O(mn \log n)$, instead of $O(n^2 \log n)$ in the full matrix case.

Algorithm 6 Computation of the decomposition (42) for $c(x,y) = h(d(x,y))$.

function COMPUTE-DECOMPOSITION($X_n, m, \varepsilon_1, c_0$)
  $\boldsymbol{D}_{n\times m} \leftarrow$ DISTANCE-CALCULATION(X_n, m)
  $[a_1, \ldots, a_{m^2}] \leftarrow \text{Sort}(\boldsymbol{D}_{m\times m})$
  // Compute the median value
  $\varepsilon \leftarrow \varepsilon_1 h(a_{\lfloor m^2/2 \rfloor})$
  // Compute the value of λ_0
  $\lambda_0 \leftarrow h(a_{\lfloor c_0 m^3/n \rfloor})$
  Compute $\hat{\boldsymbol{D}}_{n\times n}$ using λ_0
  for $1 \le i \le n,\ 1 \le j \le m$ **do**
    $r_{ij} \leftarrow \exp(-h_1(d_{ij})/\varepsilon)$
  $\alpha \leftarrow \exp\left(\frac{1}{\varepsilon mn} \sum_{1\le i\le n,\ 1\le j\le m} \left(h(d_{ij}) + \varepsilon \ln \sum_{k=1}^{m} r_{ik} r_{jk}\right)\right)$
  $\boldsymbol{S} \leftarrow \boldsymbol{0}_{n\times n}$
  for $\hat{d}_{ij} \neq 0$ **do**
    $s_{ij} \leftarrow \left(\exp\left(-h(\hat{d}_{ij})/\varepsilon\right) - \alpha \sum_{k=1}^{m} r_{ik} r_{jk}\right)$
  return $\boldsymbol{R}, \boldsymbol{S}, \alpha$

7. Experiments. In this section we compare the following approaches to approximate the optimal transport distance $W_{\boldsymbol{C}}$:

- Solve the discretization (2) of the original optimal transport problem, as a linear program.
- Bregman projections with a full distance-based kernel [3, 10].
- Bregman projections with a low-rank kernel using Algorithm 5 with (36).
- Convolutional Wasserstein distance [30]. This approach is valid for d^2 cost function.
- Fast EMD (the method from [22]). This approach is valid for $c(x,y) = \|x-y\|_1$ or $c(x,y) = \|x-y\|_2$ on 2-dimensional grids.

Both Bregman projections algorithms are implemented in MATLAB and were run with tolerance 10^{-7}. We use the notion $\varepsilon = \varepsilon_1 \,\text{median}(\boldsymbol{C})$. The linear program is solved using state-of-the-art parallel optimization software MOSEK [24]. All tests were conducted on a 3.3 GHz i5 Intel CPU with 23.5 GB RAM. The GPU tests were conducted on Nvidia GeForce GTX 960 GPU with 4GB of device memory.

7.1. Optimal transport distance test. In the following test we use an 80×80 grid and the measures shown in Figure 1.

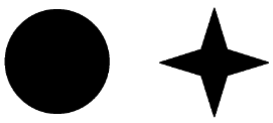

FIG. 1. *The measures used are uniform and support = black pixels.*

We test the algorithm with the cost functions $c(x,y) = \|x-y\|^p$, where $p = 1, 1.5, 2, 2.5$ and $\varepsilon_1 = 0.01, 0.02, 0.1$. We use $m = 200$ and $c_0 = 0.9$. Given an optimal

TABLE 1
(Top) Comparison of the Bregman projections results and the results of the low-rank version of the algorithm (16). *(Bottom) The linear programming results.*

	$V_{\tilde{\boldsymbol{C}}}$	$V_{\boldsymbol{C}}$	$t_{\tilde{\boldsymbol{C}}}$	$t_{\boldsymbol{C}}$
$\varepsilon_1 = 0.01, p = 1$	7.5506	7.4881	2.1311s	19.5672s
$\varepsilon_1 = 0.02, p = 1$	7.8877	7.8736	0.8146s	6.4046s
$\varepsilon_1 = 0.1, p = 1$	11.0979	11.4922	0.1007s	0.8544s
$\varepsilon_1 = 0.01, p = 1.5$	25.0549	24.9636	1.3127s	10.0903s
$\varepsilon_1 = 0.02, p = 1.5$	27.4237	27.5490	0.5636s	4.8962s
$\varepsilon_1 = 0.1, p = 1.5$	47.0038	47.3090	0.1023s	0.9417s
$\varepsilon_1 = 0.01, p = 2$	88.7902	87.7704	0.9227s	7.2519s
$\varepsilon_1 = 0.02, p = 2$	104.3581	103.9571	0.4561s	3.3858s
$\varepsilon_1 = 0.1, p = 2$	213.5207	213.1064	0.0867s	0.7878s
$\varepsilon_1 = 0.01, p = 2.5$	318.1702	335.2035	0.8668s	5.5313s
$\varepsilon_1 = 0.02, p = 2.5$	441.3424	429.4529	0.3161s	2.9220s
$\varepsilon_1 = 0.1, p = 2.5$	994.5078	1029.5	0.087s	0.8432s

	$V_{\boldsymbol{C}}(\boldsymbol{\pi})$	Time
Linear programming, $p = 1$	7.22	752.71s
Linear programming, $p = 1.5$	22.41	814.82s
Linear programming, $p = 2$	70.82	941.75s
Linear programming, $p = 2.5$	228.68	855.94s

plan $\boldsymbol{\pi}$ for the problem (4), we define

$$V_{\boldsymbol{C}} := \sum_{1 \le i,j \le n} a_i a_j c_{ij} \pi_{ij}, \qquad \boldsymbol{\pi} = \operatorname{argmin}_{\boldsymbol{\pi}} W_{\boldsymbol{C}}(\boldsymbol{\pi}). \tag{43}$$

In Table 1, we compare the optimal transport values $V_{\boldsymbol{C}}(\boldsymbol{\pi})$ for the full kernel and $V_{\tilde{\boldsymbol{C}}}(\boldsymbol{\pi})$ for the low-rank approximation. For convenience we use $a_i = 1,\ i = 1, \ldots, n$.

7.2. 2-dimensional barycenter test.

7.2.1. Thresholding test. In the following test we compute the barycenter of the following measures (see Figure 2), defined on a 140×140 grid. We test the algorithm for $p = 2$, $m = 1000$, $\varepsilon_1 = 0.01$, and different c_0 values.

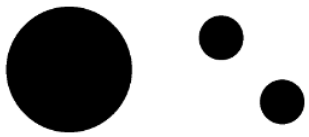

FIG. 2. *The measures used in the barycenter test.*

TABLE 2
Execution time on CPU and GPU.

	Time CPU	Time GPU	f_{CPU}	f_{GPU}	Accuracy (L_2 norm)
Low-rank with $c_0 = 0$	3.65s	0.52s	7.82	8.25	4.62e-04
Low-rank with $c_0 = 0.5$	5.21s	0.67s	5.48	6.4	2.73e-04
Low-rank with $c_0 = 0.9$	6.57s	0.799s	4.347	5.37	1.78e-04
Full-rank	28.56s	4.29s	1	1	0

The execution time is given in Table 2, where f_{CPU} and f_{GPU} are the acceleration

factors on CPU and GPU, respectively. The accuracy is the L_2 error between the low-rank barycenter and the full kernel barycenter. The accuracy is given by

$$\|(\boldsymbol{p}_{fullrank} - \boldsymbol{p}_{lowrank}) \odot \boldsymbol{a}\|_2. \tag{44}$$

We conclude that our approximation is faster than Bregman projections with full kernel. In addition, we see from Table 2 and Figure 3 that the sparse $\boldsymbol{S}$ matrix can improve the accuracy, at a computational price, but still with substantial acceleration.

FIG. 3. *Left to right: Low-rank barycenters with* $c_0 = 0$, $c_0 = 0.5$, *and* $c_0 = 0.9$ *and the full kernel barycenter.*

7.2.2. 4 figures barycenter test. In Figures 4, 5, and 6 we compute barycenters between 4 figures (140×140 images) for $p = 1, 2, 2.5$, $c_0 = 0.9$, $m = 1000$.

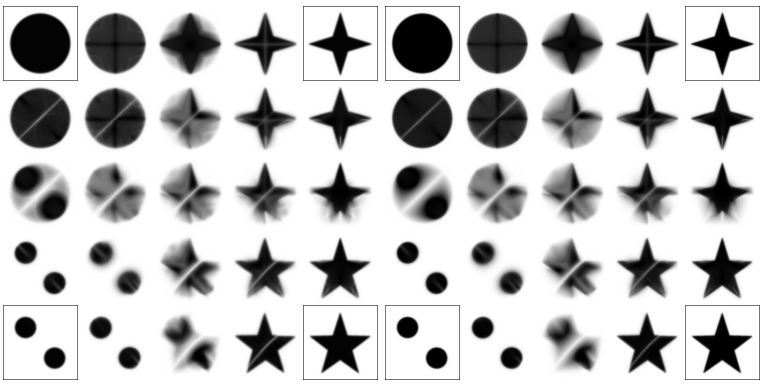

FIG. 4. *Left: Low-rank with* $p = 1$*; average time GPU* 5.5789. *Right: Full kernel with* $p = 1$*; average time* 24.6366.

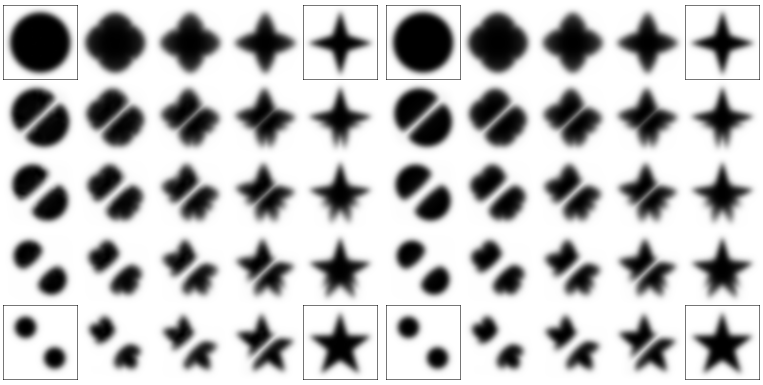

FIG. 5. *Left: Low-rank with* $p = 2$*; average time GPU* 3.0836. *Right: Full kernel with* $p = 2$*; average time* 13.2927.

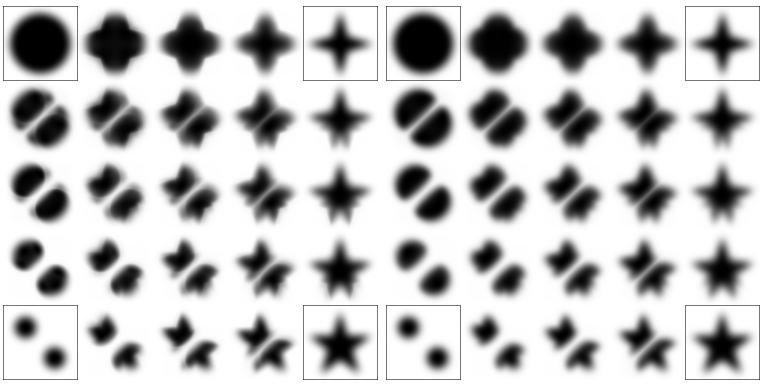

FIG. 6. *Left: Low-rank with* $p = 2.5$*; average time GPU* 2.7209. *Right: Full kernel with* $p = 2.5$*; average time* 9.7155.

7.3. Triangular domain test. In this section we include tests on triangular meshes. The following test is conducted on the cat1 mesh from the TOSCA database [7]. Due to GPU memory limitations, the mesh was downsampled to 19269 vertices and 38500 faces. In this test we calculate the barycenter of the two measures shown in Figure 7. We use $\varepsilon_1 = 0.01$ and $m = 1000$.

TABLE 3
Execution time on CPU and GPU.

	Time CPU	Time GPU	Accuracy (L_2 norm)
Low-rank with $c_0 = 0$	18.7990s	2.4986s	0.0146
Low-rank with $c_0 = 0.9$	40.3225s	4.6231s	0.0146
Full kernel	126.9557s	22.8325s	0

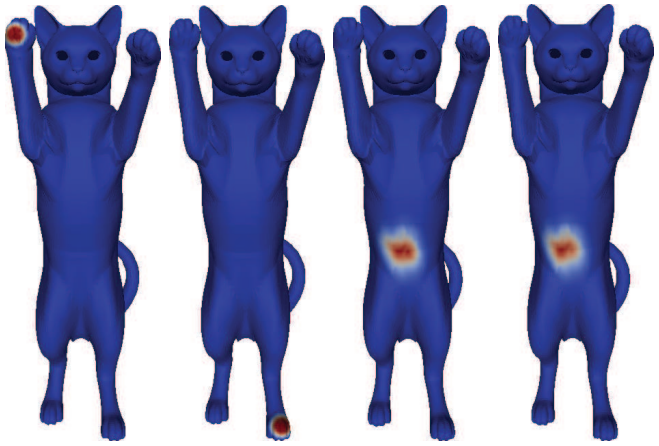

FIG. 7. *Measure* 1*, measure* 2*, low-rank barycenter, and the full kernel barycenter.*

It can be seen that thresholding is optional in this example since the transport plan is not using small distances. Hence we have the same accuracy in both cases in Table 3.

7.4. Comparison to other approaches. In this section we compare our approach to [22] in the L_2 ($p = 1$) and L_1 cases and to [30] in the case when $p = 2$. These approaches do not require computing the cost function.

7.4.1. Convolutional Wasserstein distance. In this section we compare our method to [30] for the d^2 cost function. This method is based on the Varadhan formula

$$d(x, y) = \lim_{t \to 0}(-2t \ln H_t(x, y)), \tag{45}$$

where $H_t(x, y)$ is the solution of the heat equation $\frac{\partial f}{\partial t} = \nabla f$. For $t = \varepsilon/2$ we obtain the approximation

$$e^{-d(x,y)^2/\varepsilon} \approx H_{\varepsilon/2}(x, y).$$

Solving the diffusion equation via an implicit Euler integration with time step $t = \varepsilon/2$, we obtain

$$\boldsymbol{w} = H_t(\boldsymbol{a} \odot \boldsymbol{v}) \quad \Longleftrightarrow \quad \left(\boldsymbol{D}_{\boldsymbol{a}} + \frac{\varepsilon}{2}\boldsymbol{L}\right)\boldsymbol{w} = \boldsymbol{a} \odot \boldsymbol{v}, \tag{46}$$

where $\boldsymbol{L} \in \mathbb{R}^{n \times n}$ is the cotangent Laplacian. As stated in [9], this approximation is not accurate, and the authors suggest an improvement.

In the following test, we use similar measures to those in section 7.3 in the downsampled cat mesh (6000 vertices and 12000 faces). We use $\varepsilon_1 = 0.01$, $m = 300$, and $c_0 = 0.9$.

We see in Figure 8 that the values of the optimal transport value diverge after several iterations. This happens even if we use a very small ε_1 value. The second example is using $\varepsilon_1 = 0.0001$ and is calculated using the Multiprecision Computing Toolbox for MATLAB [1].

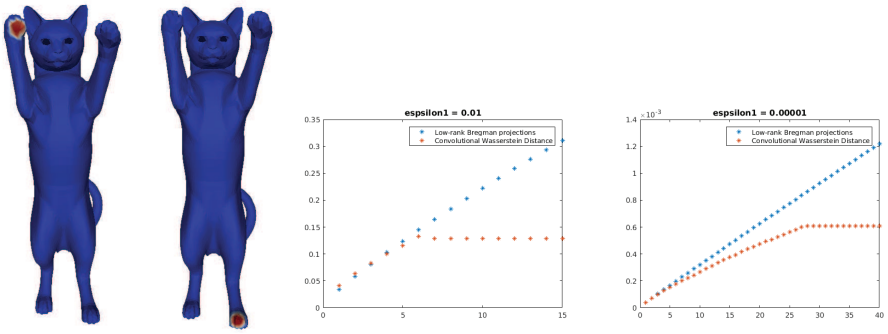

FIG. 8. *Measure* 1, *measure* 2, *first Bergman projections iterations.*

TABLE 4
Numerical results.

	Time CPU	Result
Full kernel Bregman	10.5672s	0.9520
Low-rank Bregman with $\varepsilon_1 = 0.01$ and $c_0 = 0.9$	2.0789s	0.9599
Convolutional Wasserstein distance	0.4609s	0.1291
Linear programming	1962.06s	0.9427

From Table 4 we conclude that our approach, although it requires computing a portion of the distance matrix, is comparable numerically to the real optimal transport problem. The convolutional Wasserstein approach is faster, but it is not accurate.

It is worth noting that the authors in [30] use the convolutional Wasserstein approach successfully for the Wasserstein barycenter problem, using a modified barycenter problem and introducing the "entropic sharpening technique."

7.4.2. Fast EMD. In this subsection we discuss the approach [22] for solving the optimal transport problem. This approach works for the Euclidean distance $c(x, y) = \|x - y\|$ and for the L_1 distance $c(x, y) = \|x - y\|_1$. Both cost functions can be approximated using the semidiscrete approximation (the $p = 1$ case in (8) and (9)). We call this approach Fast EMD.

Let $L(v)$ be homogeneous of degree 1 and convex in v. For example, $L(v) = \|v\|_2$ yields the Euclidean distance and $L(v) = \|v\|_1$ the Manhattan distance. The idea is that for a cost function of the form

$$c(x, y) = \min \left\{ \int_0^1 L(\dot{\gamma}(t))dt; \;\; \gamma(0) = x, \;\; \gamma(1) = y, \;\; \gamma \in C^1[0, 1] \right\}$$

the optimal transport problem is formulated as

$$\begin{aligned} &\text{minimize} \int_\Omega L(m(x))dx \\ &\text{subject to } \nabla m(x) + \rho_1(x) - \rho_0(x) = 0, \\ &m(x) \cdot n(x) = 0 \text{ for all } x \in \partial\Omega \text{ such that } n(x) \text{ is normal to } \partial\Omega. \end{aligned}$$

The authors concentrate on the 2-dimensional grid case. In the L_1 case, the discretization gives the problem

$$\text{minimize}\|\boldsymbol{m}\|_{1,1} \text{ subject to } \operatorname{div}(\boldsymbol{m}) + \rho_1 - \rho_0 = 0.$$

The algorithm to compute the optimal transport has the form

$$\begin{aligned}\boldsymbol{m}^{k+1} &= \operatorname{shrink}(\boldsymbol{m}^k + \mu\nabla\Phi_k),\\ \Phi^{k+1} &= \Phi_k + \tau(\operatorname{div}(2\boldsymbol{m}^{k+1} - \boldsymbol{m}^k) + \rho_1 - \rho_2).\end{aligned}$$

Here $\mu, \tau > 0$ are the algorithm's parameters, ∇, div are discrete gradient and divergence operators, respectively, and the shrink operator is a function that depends on the ground metric. A similar algorithm can be obtained in the L_2 case. Under appropriate conditions, it is proven in [22] that $\boldsymbol{m}_k$ converge to the solution of the optimal transport problem.

We use the same example as in section 7.1 downsampled to a 64×64 grid on $[0,1] \times [0,1]$. We use a MATLAB implementation and a CUDA C++ implementation provided by the authors of [22]. The parameters of the algorithm are $\tau = 1$ and $\mu = 1.5747e-05$, $\Delta x = 63$, 1000 iterations. Since the Fast EMD algorithm gives an exact optimal transport value, we compare only the value $V_{\boldsymbol{C}}$ of the entropic regularized algorithms ((43) with $a_i = 1$ for all i). For the low-rank Bregman projections algorithm we use $m = 200$, $c_0 = 0.9$, $\varepsilon_1 = 0.01$, and tolerance 10^{-5}.

TABLE 5
Comparison of the Bregman projections algorithm (full kernel and low-rank kernel) with the Fast EMD approach.

	Result	Time CPU	Time GPU
Fast EMD (L_2)	0.0900	0.37898s	0.01932s
Bregman projections L_2, low rank	0.09324	0.18235s	0.04224s
Bregman projections L_2, full kernel	0.09235	1.27196s	0.19515s
Linear programming (L_2)	0.090364	186.66s	-
Fast EMD (L_1)	0.1231	0.24666s	0.028720s
Bregman projections L_1, low rank	0.125808	0.16454s	0.05473s
Bregman projections L_1, full kernel	0.12596	0.97466s	0.05052s
Linear programming (L_1)	0.12727	162.71s	-

From Table 5 we see that the results are comparable to ours. The dynamical approach has simple implementation, similarly to the Bregman projections algorithm, and it also can be parallelized. The advantage of using Fast EMD is that this approach does not require one to compute distances. The main disadvantage, as pointed in [22], is the problem of choosing the correct μ and τ.

8. Conclusions. In this paper, we develop a method to accelerate the Bregman projection algorithm for solving the entropic regularized optimal transport problem and related problems on Riemannian manifolds. Our method is valid for a large family of cost functions, including the d^p cost functions where $p \geq 1$. We propose using a low-rank matrix decomposition to approximate the matrix $e^{-\boldsymbol{C}/\varepsilon}$. This approximation has two advantages. First, it requires the calculation of only a small portion of the geodesic distances matrix. Second, it reduces the complexity of each iteration in the Bregman projection algorithm.

We develop two decompositions which are based on the semidiscrete approximation. The first decomposition is obtained directly by smoothing the semidiscrete

approximation. In the second decomposition we add parameters to the first decomposition and optimize them to fit the known distances. In addition we calculate the small distances explicitly and use thresholding to make the cost entries associated with small distances more accurate. This improves the accuracy of the approximation at the expense of additional preprocessing time.

Experimental results show that both approximations are substantially faster than the original Bregman projection algorithm. While the second decomposition is more accurate in the general case, the first decomposition can be used for problems in which the transport plan does not use small distances (such as the case of two measures with disjoint supports). Although we applied the decompositions only for accelerating the computation of optimal transport distances and Wasserstein barycenters, they can be used in other optimal transport related problems in the fields of image processing and machine learning.

Appendix A. Approximation errors. In this appendix we compare the approximation errors of the different decompositions considered in this paper (such as (21) or (35)). First, we describe a method to estimate the approximation error. A common way to do this (see [26]) is to use the average of the elementwise relative errors,

$$error^{\boldsymbol{C},\tilde{\boldsymbol{C}}} := \frac{1}{n^2} \sum_{1 \le i,j \le n} e_{i,j}^{\boldsymbol{C},\tilde{\boldsymbol{C}}}, \tag{47}$$

where

$$e_{ij}^{\boldsymbol{C},\tilde{\boldsymbol{C}}} := \left| \frac{c_{ij} - \tilde{c}_{ij}}{c_{ij}} \right|, \quad 1 \le i,j \le n. \tag{48}$$

We need to exclude the cases $c_{ij} = 0$ in (47) or to add $\epsilon > 0$ to the denominator of (48). We use a more subtle method in order to better capture the quality of the approximation. We divide the elements of $\tilde{\boldsymbol{C}}$ into disjoint equal cardinality subsets and compute the average relative error in each subset. Given a set $F \subset \{c_{ij}\}_{1 \le i,j \le n}$ we define the average error on F,

$$f(F) := \frac{1}{|F|} \sum_{c_{ij} \in F} e_{i,j}^{\boldsymbol{C},\tilde{\boldsymbol{C}}}. \tag{49}$$

We define the equal-size subsets by

$$F_{i_0} = \{c_{ij} \mid a_{i_0 - 1} < c_{ij} \le a_{i_0}\}, \quad 1 \le i_0 \le l, \tag{50}$$

where

$$0 = a_0 < a_1 < \cdots < a_l = \max_{1 \le i,j \le n} c_{ij}$$

are such that $|F_i| = |F_j|$, $1 \le i,j \le n$. Then the error is represented as a vector of errors on each subset:

$$error_l^{\boldsymbol{C},\tilde{\boldsymbol{C}}} := (F_1, \ldots, F_l). \tag{51}$$

We see that the elements such that $c_{ij} = 0$ are excluded by definition.

All the tests in this appendix are conducted for the duck mesh (Figure 9) and the d^2 cost function.

A.1. Semidiscrete approximations. In the following figure we use this methodology to compare the errors of the semidiscrete approximation $c_{ij}^{Z_m}$, the regularized semidiscrete approximation $c_{ij}^{Z_m,\varepsilon}$ (21), and the two versions (35), (42) of the optimized semidiscrete approximation.

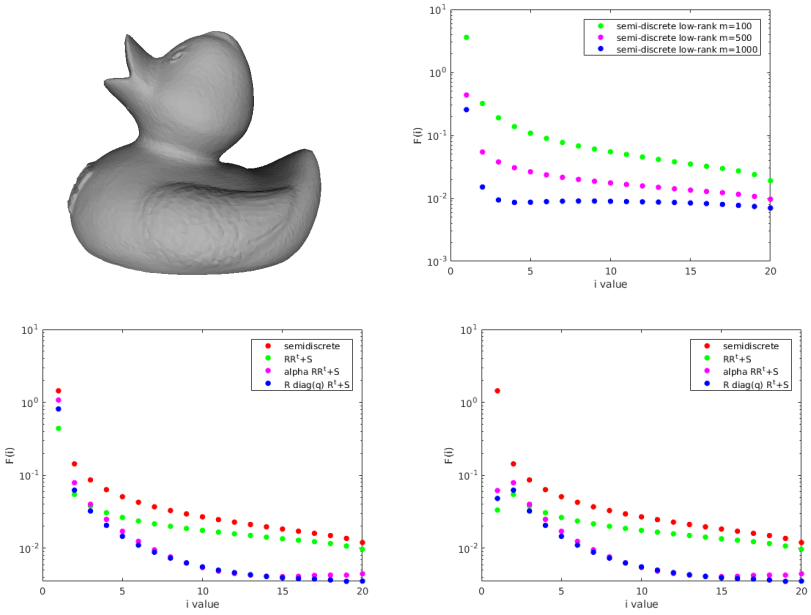

FIG. 9. *Top left: The duck mesh. Top right: The relative errors of the low-rank approximation* (21) *for different values of* m. *Bottom left: Comparison of the approximations* $\boldsymbol{R}\boldsymbol{R}^t$, $\alpha\boldsymbol{R}\boldsymbol{R}^t$, *and* $\boldsymbol{R}\,\mathrm{diag}(\boldsymbol{q})\boldsymbol{R}^t$, $m = 500$, $\varepsilon_1 = 0.01$ *on the duck mesh. Bottom right: Comparison of the approximations with thresholding. The y-axis is logarithmic in these charts.*

In Figure 10 we compare $\lambda(m)$ with the estimated value in (18) (with a specific C value) for different m values.

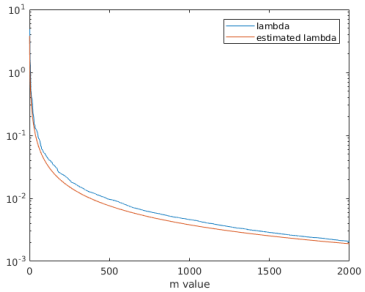

FIG. 10. *Comparison between* λ *and the estimated value* (18) *on the duck mesh,* $C = \frac{1}{\sqrt{2}}$, *and* $\dim(X) = 2$.

A.2. The Nystrom low-rank decomposition. In this section we present numerical experiments regarding the approximation of $\boldsymbol{C} = \{d_{ij}^2\}$ and $\boldsymbol{H}(\boldsymbol{C},\varepsilon) = \exp(-\boldsymbol{C}/\varepsilon)$, using the Nystrom method. It can be seen in Figure 11 that as ε_1 gets

smaller the approximation becomes inaccurate at large distances. This prevents the use of the Nystrom method in the Bregman projections algorithm, where ε_1 is required to be very small; a typical choice is $\varepsilon_1 = 0.01$.

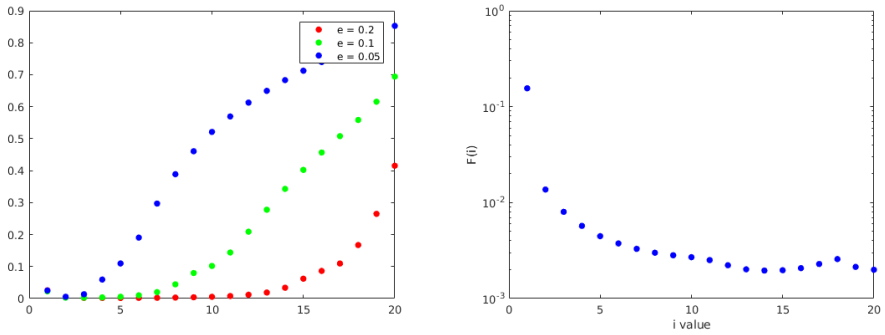

FIG. 11. *Left: The relative errors of the Nystrom approximation of* $\boldsymbol{H}(\boldsymbol{C}, \varepsilon)$ *for* $\varepsilon_1 = 0.05, 0.1, 0.2$. *Right: Nystrom approximation of* $\boldsymbol{C}$.

REFERENCES

[1] ADVANPIX LLC, *Multiprecision Computing Toolbox for MATLAB, version* 4.4.7.12739, Tokyo, Japan, 2017, https://www.advanpix.com/.

[2] M. AGUEH AND G. CARLIER, *Barycenters in the Wasserstein space*, SIAM J. Math. Anal., 43 (2011), pp. 904–924, https://doi.org/10.1137/100805741.

[3] J.-D. BENAMOU, G. CARLIER, M. CUTURI, L. NENNA, AND G. PEYRÉ, *Iterative Bregman projections for regularized transportation problems*, SIAM J. Sci. Comput., 37 (2015), pp. A1111–A1138, https://doi.org/10.1137/141000439.

[4] J.-D. BENAMOU, B. D. FROESE, AND A. M. OBERMAN, *Numerical solution of the optimal transportation problem using the Monge-Ampère equation*, J. Comput. Phys., 260 (2014), pp. 107–126.

[5] J.-D. BENAMOU, Y, BRENIER, AND K. GUITTET, *The Monge–Kantorovich mass transfer and its computational fluid mechanics formulation*, Internat. J. Numer. Methods Fluids, 40 (2002), pp. 21–30.

[6] L. M. BREGMAN, *The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming*, USSR Comput. Math. Math. Phys., 7 (1967), pp. 620–631 (in Russian).

[7] A. M. BRONSTEIN, M. M. BRONSTEIN, AND R. KIMMEL, *Numerical Geometry of Non-Rigid Shapes*, Springer, New York, 2008.

[8] M. CHEN AND M. CHIANG, *Distributed optimization in networking: Recent advances in combinatorial and robust formulations*, in Modeling and Optimization: Theory and Applications, Springer, New York, 2012, pp. 25–52.

[9] K. CRANE, C. WEISCHEDEL, AND M. WARDETZKY, *Geodesics in heat: A new approach to computing distance based on heat flow*, ACM Trans. Graphics, 32 (2013), 152.

[10] M. CUTURI AND A. DOUCET, *Fast computation of Wasserstein barycenters*, Proc. Mach. Learn. Res., 32 (2014), pp. 685–693.

[11] M. CUTURI, *Sinkhorn distances: Lightspeed computation of optimal transport*, in Proceedings of the 26th International Conference on Neural Information Processing Systems Volume 2, Lake Tahoe, NV, 2013, pp. 2292–2300.

[12] E. W. DIJKSTRA, *A note on two problems in connexion with graphs*, Numer. Math., 1 (1959), pp. 269–271.

[13] P. DRINEAS AND M. W. MAHONEY, *On the Nystrom method for approximating a Gram matrix for improved kernel-based learning*, J. Mach. Learn. Res., 6 (2005), pp. 2153–2175.

[14] S. HAKER, L. ZHU, A. TANNENBAUM, AND S. ANGENENT, *Optimal mass transport for registration and warping*, Internat. J. Comput. Vis., 60 (2004), pp. 225–240.

[15] D. S. HOCHBAUM AND D. B. SHMOYS, *A best possible heuristic for the k-center problem*, Math. Oper. Res., 10 (1985), pp. 180–184.

[16] L. V. Kantorovich, *On the translocation of masses*, C. R. (Doklady) Acad. Sci. Nauk URSS (N.S.), 37 (1942), pp. 199–201.
[17] Y. H. Kim and B. Pass, *Multi-marginal optimal transport on Riemannian manifolds*, Amer. J. Math., 137 (2015), pp. 1045–1060.
[18] R. Kimmel and J. A. Sethian, *Computing geodesic paths on manifolds*, Proc. Natl. Acad. Sci. USA, 95 (1998), pp. 8431–8435.
[19] C. Leonard, *From the Schrödinger problem to the Monge-Kantorovich problem*, J. Funct. Anal., 262 (2012), pp. 1879–1920.
[20] B. Lévy, *A numerical algorithm for L_2 semi-discrete optimal transport in 3D*, ESAIM Math. Model. Numer. Anal., 49 (2015), pp. 1693–1715.
[21] W. Li, P. Yin, and S. Osher, *Computations of optimal transport distance with Fisher information regularization*, J. Sci. Comput., 75 (2018), pp. 1581–1595.
[22] W. Li, E. K. Ryu, S. Osher, W. Yin, and W. Gangbo, *A parallel method for earth mover's distance*, J. Sci. Comput., 75 (2018), pp. 182–197.
[23] Q. Merigot, *A multiscale approach to optimal transport*, Comput. Graphics Forum, 30 (2011), pp. 1583–1592.
[24] MOSEK ApS, *MOSEK version* 8, 2017, https://mosek.com.
[25] E. K. Ryu, W. Li, P. Yin, and S. Osher, *Unbalanced and partial L_1 Monge-Kantorovich problem: A scalable parallel first-order method*, J. Sci. Comput., 75 (2018), pp. 1596–1613.
[26] G. Shamai, Y. Aflalo, M. Zibulevsky, and R. Kimmel, *Classical scaling revisited*, in Proceedings of the IEEE International Conference on Computer Vision (ICCV), IEEE, Washington, DC, 2015.
[27] G. Shamai, M. Zibulevsky, and R. Kimmel, *Accelerating the computation of canonical forms for 3D nonrigid objects using multidimensional scaling*, in Proceedings of the 2015 Eurographics Workshop on 3D Object Retrieval, Eurographics Association, Aire-la-Ville, Switzerland, 2015.
[28] R. Sinkhorn, *Diagonal equivalence to matrices with prescribed row and column sums*, Amer. Math. Monthly, 74 (1967), pp. 402–405.
[30] J. Solomon, F. De Goes, G. Peyre, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Trans. Graphics, 34 (2015), 66.
[31] C. Villani, *Topics in Optimal Transportation*, Grad. Stud. Math. 58, AMS, Providence, RI, 2003.
[32] C. Villani, *Optimal Transport: Old and New*, Springer, New York, 2009.
[33] G. Wolansky, *On semi-discrete Monge-Kantorovich and generalized partitions*, J. Optim. Theory Appl., 165 (2015), pp. 359–384.