Computational Optimal Transport: A Comparison of Two Algorithms

Benoît MÜLLER

École Polytechnique Fédérale de Lausanne
Semester project, applied mathematics

Supervisor: Lénaïc Chizat
Co-supervisor: Tomas Vaškevičius

2023

Abstract

We present here two formulations of the Monge optimal transport problem. The first one is a discretization leading to the linear sum assignment problem, which we solve with the Hungarian algorithm. The second is the dynamical formulation of Benamou-Brenier and leads to a functional saddle problem, which we solve by an augmented Lagrangian method. Both methods are implemented, numerically analyzed, and compared.

Contents

0.1 Introduction . . . 1
0.1.1 Contributions . . . 1
0.1.2 The problem . . . 1
0.1.3 Theoretical background . . . 2
0.2 Discrete formulation . . . 3
0.2.1 Discretization of measure leads to linear programming . . . 3
0.2.2 Duality and KKT conditions . . . 4
0.2.3 Hungarian algorithm . . . 5
0.2.4 Numerical results . . . 9
0.3 Dynamical formulation . . . 11
0.3.1 Benamou Brenier formula . . . 11
0.3.2 Augmented Lagrangian optimization . . . 14
0.3.3 Numerical results . . . 17
0.4 Comparison of the two formulations . . . 19
0.5 Proofs . . . 22

0.1 Introduction

0.1.1 Contributions

In this project, we have assembled the steps to prove that the Monge problem is equivalent to the Linear sum assignment problem, and we have done the same for the dynamical formulation. The Hungarian algorithm has been implemented in his $O(n^3)$ version and tested numerically to exhibit a better practical complexity of $O(n^{2.65})$. For the dynamical formulation of Benamou Brenier, we proposed to solve the Poisson equation of the primal step with finite differences and a stored factorization of the stiffness matrix. This showed better complexity with respect to the number of iterations, almost linear, and accelerated the running time. It didn't change the complexity with respect to the discretization, but in practice, we observed a better local rate and running time. Our implementation allowed us to reproduce the results of Benamou and Brenier in the original paper on the dynamical method.

0.1.2 The problem

The problem of optimal transport started with Gaspard Monge in 1781, where he formulated the problem of transporting a distributed mass to another one with a minimum possible cost.

The two masses are now represented by probabilities measures μ,ν on some spaces $\mathcal{X}$,$\mathcal{Y}$. Since we have probabilities measures, we can interpret the mass by two random variables X,Y that

have their distributions. The transport itself is represented by a transport map T that needs to be compatible: $T(X)$ has the same distribution of Y:

$$\nu(B) = \mu(T^{-1}(B)) \text{ for any Borel set } B \text{ of } \mathcal{X}.$$

In this case we say that ν is the pushforward of μ and we write $T_{\#}\mu = \nu$. The cost is determined by a function $c : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ defined on any pair of points, and by averaging on the space we define the cost of transport:

$$\mathbb{E}[c(X, T(X))] = \int c(x, T(x)) \, \mathrm{d}\mu(x).$$

Typical costs are $c(x, y) = \|x - y\|_p^p$, and in our project, we will stay in the quadratic case $p = 2$ for its regularity and properties that are shown in the next section. The Monge problem (MP) is hence

$$\inf_{T_{\#}\mu=\nu} \int \|x, T(x)\|^2 \, \mathrm{d}\mu(x) \tag{MP}$$

0.1.3 Theoretical background

The Monge problem gained a bit more interest when Kantorovich defined a relaxation of the problem, where each point of $\mathcal{X}$ can be linked to more than one point of $\mathcal{Y}$. This is defined by a probability measure γ on $\mathcal{X} \times \mathcal{Y}$, with the compatibility condition that the total mass sent from a set must be its measure:

$$\gamma(A \times \mathcal{Y}) = \mu(A)$$

, and the total mass received by a set must be its measure:

$$\gamma(\mathcal{X} \times B) = \nu(B).$$

In other words, the measures μ and ν are the marginal distributions of γ which is then called a coupling and we write the set of couplings

$$\mathcal{P}(\mu, \nu) = \{\gamma \text{ Borel measure on } \mathcal{X} \times \mathcal{Y} | \pi_{1\#}\gamma = \mu, \, \pi_{2\#}\gamma = \nu\}.$$

This leads to a relaxed version of (MP), called the Kantorovich problem:

$$\inf_{\gamma \in \mathcal{P}(\mu,\nu)} \int \|x, y\|^2 \, \mathrm{d}\gamma(x, y) \tag{KP}$$

every transport map T gives the same cost as the coupling $(Id \times T)_{\#}\mu$ so (KP) is a lower bound of (MP): (KP)$\leq$(MP). The set $\mathcal{P}(\mu, \nu)$ has the good property to be non a non-empty convex set, and the objective function is linear. These are the ingredients that bring duality theory. The dual formulation of (KP) will be central in both algorithms presented later so we present it here:

Theorem 1 (Kantorovich duality)**.** *Let μ and ν be measure on $\mathbb{R}^d$ with finite first moments, then (KP) admit a dual formulation with dual variables $\phi, \psi : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ that are measurable maps:*

$$\min_{\gamma \in \mathcal{P}(\mu,\nu)} \int -x \cdot y \, \mathrm{d}\gamma(x, y) = \max_{\phi(x)+\psi(y) \leq -x \cdot y} \int \phi(x) \, \mathrm{d}\mu(x) + \int \psi(x) \, \mathrm{d}\nu(y) \tag{KD}$$

Proof. See [10, Theorem 1.3]. □

The pertinence of the Kantorovich formulation comes from the fact that under some regularity conditions, both problems are actually equivalent:

Theorem 2 (Brenier's Theorem)**.** *Let μ and ν be measures on $\mathbb{R}^d$ with finite second moments and such that μ is absolutely continuous w.r.t. the Lebesgue measure.*

Then (KP) has a μ-unique [1] optimal coupling γ. In addition, $\gamma = (Id \times T)_{\#}\mu$ and $T = \nabla\phi$ for a convex function ϕ.

Proof. [10, Theorem 2.12] □

In other words, the solution of (KP) is actually a transport map and (MP)=(KP). Furthermore, we can show that the convexity of ϕ is actually necessary and sufficient:

Corollary 1. *Under the hypothesis of Theorem 2, there exists an μ-unique optimal transport map. In addition, a transport map T is optimal if and only if $T = \nabla\phi$ μ-a.e. for a convex function ϕ.*

Proof. [10, Theorem 2.12] □

We can use transport as a notion of distance between measures:

Theorem 3 (Wasserstein's distance)**.** *$W_2(\mu, \nu) = \sqrt{(KP)}$ is a distance on the space of probability measures with finite second moments.*

Proof. [10, Theorem 7.3] □

We know that the solution of (MP) exists, is unique, and since a distance is continuous, the problem of finding the cost for given measures is well posed in the sense of Hadamard. The better we approximate the measures, the better we approximate the cost.

0.2 Discrete formulation

0.2.1 Discretization of measure leads to linear programming

Suppose we can approximate the measures with Dirac measures in the following way:

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i}, \quad \nu = \frac{1}{n}\sum_{j=1}^{n}\delta_{y_j},$$

for n pairs of points $x_i, y_j \in \mathbb{R}^d$. Then we can show that the problem is actually equivalent to a linear program.

We begin to see that the set of transport maps can be more precisely determined.

Lemma 1. *The set of transport maps from $\mu = \frac{1}{n}\sum_{i=1}^{n}\delta_{x_i}$ to $\nu = \frac{1}{n}\sum_{j=1}^{n}\delta_{y_j}$ is*

$$\{T : \mathbb{R}^d \to \mathbb{R}^d | \exists\sigma \in S_n : T(x_i) = y_{\sigma(i)} \forall i \in \{1, \dots, n\}\},$$

i.e. T only need to permute $\{x_i\}_{i=1}^n$ and $\{y_j\}_{i=1}^n$.

Proof. See Page 22 in the proof Section 0.5. □

With this understanding of the discrete transport maps, we can reformulate the problem like follows:

[1] the value of T is unique on the support of μ, outside the value can be set arbitrarily

Lemma 2. *The Monge problem is equivalent to the linear sum assignment problem (LSAP), which is an integer linear program:*

$$\min_{T_{\#}\mu=\nu} \int_{\mathbb{R}^d} \|x - T(x)\|^2 \, \mathrm{d}\mu = \min_{\substack{P_{ij}\in\{0,1\} \\ P\mathbb{1}=\mathbb{1}=P^\top\mathbb{1}}} \langle P, C\rangle \tag{1}$$

with $C_{ij} = \|x_i - y_j\|^2$

Proof. See Page 23 in the proof Section 0.5. □

This formulation can be understood in terms of graphs. The cost matrix C is associated with a weighted bipartite graph, with two vertices components of the same size n, and where C_{ij} is the weight between the i-th and j-th vertex. We consider this bipartite graph as complete in the sense that all the n^2 possible edges are given a non-negative weight, possibly zero. The permutation matrix P represents a perfect matching, this is why it is called an "assignment problem". The cost associated with a matching P is $\langle P, C\rangle$, explaining the "linear sum" name.

Lemma 3. *The scalar version of LSAP (1) always has integer solutions. In particular*

$$\min_{\substack{P_{ij}\in\{0,1\} \\ P\mathbb{1}=\mathbb{1}=P^\top\mathbb{1}}} \langle P, C\rangle = \min_{\substack{P\geq 0 \\ P\mathbb{1}=\mathbb{1}=P^\top\mathbb{1}}} \langle P, C\rangle \tag{2}$$

Proof. See Page 23 in the proof Section 0.5. □

Puting Lemma 1, Lemma 2, andLemma 3 together, we conclude the following theorem:

Theorem 4. *The Monge problem is an integer solution of the following linear program on doubly stochastic matrices:*

$$\min_{T_{\#}\mu=\nu} \int_{\mathbb{R}^d} \|x - T(x)\|^2 \, \mathrm{d}\mu = \min_{\substack{P_{ij}\geq 0 \\ P\mathbb{1}=\mathbb{1}=P^\top\mathbb{1}}} \langle P, C\rangle \tag{3}$$

with $C_{ij} = \|x_i - y_j\|^2$

Proof. Direct result of Lemma 1, Lemma 2, and Lemma 3. □

As a result, we could use all the machinery of linear program solvers. However, linear programming models a very wide range of problems, and general solvers are slow (exponential). A more refined algorithm that takes into account the particular structure of this problem can have a better time complexity, like the Hungarian algorithm that we are going to use and which have cubic time complexity. We will motivate its idea by writing its dual and deriving necessary and sufficient conditions.

0.2.2 Duality and KKT conditions

Now that we have a linear programming formulation of the problem, we use the duality theory to gain some insight into the solutions. By Lemma 3 the linear problem admits an optimal solution, so a strong duality theorem for linear programming holds. We have then the two equivalent problems

$$\text{(primal)} \qquad \min_{\substack{P\geq 0 \\ P\mathbb{1}=\mathbb{1}=P^\top\mathbb{1}}} \langle P, C\rangle = \max_{\substack{u,v\in\mathbb{R}^d \\ u_i+v_j\leq C_{ij}}} \langle \mathbb{1}, u+v\rangle \qquad \text{(dual)} \tag{4}$$

Let P,u,v be feasible primal and dual variables. Then they are optimal if and only if

$$\langle P, C\rangle = \langle \mathbb{1}, u + v\rangle$$

We compute that

$$\begin{aligned}\langle P, C\rangle - \langle \mathbb{1}, u+v\rangle &= \sum_{i,j=1}^{n} P_{ij}C_{ij} - \sum_{i=1}^{n} u_i - \sum_{j=1}^{n} v_j \\ &= \sum_{i,j=1}^{n} P_{ij}C_{ij} - \sum_{i=1}^{n} u_i \sum_{j=1}^{n} P_{ij} - \sum_{j=1}^{n} v_j \sum_{i=1}^{n} P_{ij} \\ &= \sum_{i,j=1}^{n} P_{ij}(C_{ij} - u_i - v_j).\end{aligned}$$

By feasibility conditions, the terms are nonnegative, so P, u, v are optimal if and only if

$$P_{ij}(C_{ij} - u_i - v_j) = 0 \quad \forall i, j \in \{1, \dots, n\},$$

i.e. i can only be assigned to j where dual feasibility is active, this is called complementary slackness. Together with feasibility, we hence recover the KKT conditions, which we resume here:

$$\begin{cases} P \geq 0, \quad P\mathbb{1} = P^\top \mathbb{1} = \mathbb{1} & : \text{primal feasibility (pf)} \\ u_i + v_j \leq C_{ij} \quad \forall i, j \in \{1, \dots, n\} & : \text{dual feasibility (df)} \\ P_{ij}(C_{ij} - u_i - v_j) = 0 \quad \forall i, j \in \{1, \dots, n\} & : \text{complementary slackness (cs)} \end{cases} \tag{KKT}$$

0.2.3 Hungarian algorithm

The Hungarian algorithm is the first polynomial time algorithm to solve the linear sum assignment problem. It is a primal-dual algorithm that exploits the (KKT) conditions together with the fact that integer solutions exist. During the process, (df) and (cs) are always satisfied and (pf) is partially satisfied in the sense that

$$P_{ij} \in \{0, 1\}, \quad P\mathbb{1}, P^\top \mathbb{1} \leq \mathbb{1}. \tag{ppf}$$

In other words, by default, P is filled with zeros so that it satisfies (cs) and we iteratively increase the number of ones, until we have added n of them and $P\mathbb{1} = P^\top \mathbb{1} = \mathbb{1}$, assuring optimality.

The key step to always be able to fill P is a dual step, where we update the dual variables so that we can continue the filling. We organize the process in three steps:

1. Initialize P, u, v such that it satisfy (ppf),(df), and (cs).
   This can be done by various methods, either starting with $P, u, v = 0$ or a more optimal first guess.

2. Where $u_i + v_j = C_{ij}$, try to increase the number of ones in P.
   This is done by considering the bipartite sub-graph induced by zero entries of the reduced cost matrix $\tilde{C}_{ij} = C_{ij} - u_i - v_j$, and trying to find a perfect matching. For this, we grow an alternating tree to try to find an augmenting path, which we will define later.

3. When step 2. fails and we cannot increase the matching, we update u, v wisely so that we can actually continue and return to 2.

Step 1: Initialization

We use a greedy method, starting with $P, u, v = 0$. To keep $u_i + 0 \leq C_{ij}$ true, we set $u_i = \min_j C_{ij}$. Then, to keep $u_i + v_j \leq C_{ij}$ true, we set $v_j = \min_i(C_{ij} - u_i)$. Notice that with this choice, each row of $\tilde{C}$ has a zero entry in $\operatorname{argmin}_j C_{ij}$. To fill P, we iterate on the row and fill one of those zero-entry places if the column was not already taken by another row. We also store the value of P in functional style fashion, with a vector row such that

$$\begin{cases} \text{row}_j = i & \text{if } P_{ij} = 1 \\ \text{row} = \text{None} & \text{if } P_{ij} = 0. \end{cases}$$

At the end, this will give us $P_{\text{row}_j j} = 1$ and row will be the inverse of the permutation associated with P. This leads to the following algorithm:

Algorithm 1: Initialize
Compute an initial value for P, u, v that satisfies partial (KKT).

Input : $C \in \mathbb{R}^{n \times n}$
Output: A permutation matrix $P \in \mathbb{R}^{d \times d}$, and $u, v \in \mathbb{R}^n$ that satisfy partial (KKT).

$\forall i \in \{1, \dots, n\} : u_i = \min_j C_{ij}$
$\forall j \in \{1, \dots, n\} : v_j = \min_i(C_{ij} - u_i)$
$\forall i, j \in \{1, \dots, n\} : P_{ij} = 0$
for $i, j = 1, \dots, n$ **do**
  for $j = 1, \dots, n$ **do**
    if $C_{ij} - u_i - v_j = 0$ & $\text{row}_j = \textit{None}$ **then**
      $P_{ij} = 1$
      $\text{row}_j = i$
    end if
  end for
end for
return *P,u,v*

Step 2: Grow an alternating tree

As we said before, the second step consists of trying to increase the number of ones in P, keeping partial (KKT) true. As a result, we have to look only at the edges that have active (df), where the reduced cost matrix is zero. This induces a sub-graph, and we use the alternating path formulation of perfect matchings. An alternating path is a simple path such that its edges are alternatively inside and outside of the matching. An alternating path that has its first and last vertices outside the matching is called augmenting because if you change the membership of its edges to the matching, the matching increase in size by one. This characterization is actually necessary:

Theorem 5. *[3, Corollary 3.4] A matching is perfect if and only if there exists no augmenting path.*

This gives a method to augment the matching: find an augmenting path. To do this, we grow a tree formed of alternating paths, called an alternating tree.

At any iteration, a vertex is labeled if it belongs to the alternating tree. The first labeled vertex is the chosen root. The set of vertices for the row is called U, and V for the columns. At each iteration, we try to go from U to an unlabeled vertex of V using an un-assigned edge, and

then come back to U using an assigned edge. Looking for edges to come back is called scanning. At each iteration, there are three possibilities: we augment the tree, we augment the matching, or there is no vertex to scan. Suppose we are at vertex i:

- There exist some unlabeled neighbor vertices j, we label them and keep track of the tree with a predecessor vector pred. We can then scan the:
  - Augment the tree: One of them is on the matching, use its assigned edge to go back to U on a new i and label it. Set the j as scanned.
  - Augment the matching: None of them are on the matching, so we are at the end of an augmenting tree. Use pred to come back to the root and invert all edges.
- No more vertex to scan: There is no unlabeled neighbor vertex. We need to add a new edge from the labeled vertices of U to the unlabeled of V, but without taking off the ones of the alternating tree. This is done with the dual update of step 3.

Step 3: Dual update

In the Dual update, we want to solve the fail in step 2, we are stuck with an alternating tree and we fail to produce an augmenting path. We want to add a new edge from a vertex of SU to an unlabeled vertex of V, so that the tree can grow, but without changing the existing edges. Among the possibilities, we chose the one with the smallest reduced cost δ, and $\forall j \in LV$, we subtract δ from v_i. Also, $\forall i \in SU$, we add δ to u_i to balance the contributions. This only augments reduced costs of edges that were already positive but adds a new zero-reduced cost.

To implement the algorithm efficiently, we keep track of the growing tree along the process and we keep also track of the minimum reduced cost π_j of the edges connecting a labeled vertex $i \in U$ to j. The set AU denotes the assigned vertices. The pseudo-code in given in two functions Algorithm 2 and Algorithm 3

Algorithm 2: Augment
Find an augmenting tree rooted at an unassigned vertex $k \in U$

Input : C, U, V, row, k
Output: $sink, pred, U, V$

$\forall j \in V : \pi_j = \infty$ `# Chose a root for the tree`
$SU = LV = SV = \varnothing$
$sink = 0, i = k$ `# sink will be the leaf of a augmenting path`
while $sink = 0$ **do**
  $SU = SU \cup \{i\}$ `# scan i:`
  for $j \in V \setminus LV : C_{ij} - u_i - v_j < \pi_j$ **do**
    $pred_j = i$, $\pi_j = C_{ij} - u_i - v_j$ **if** $\pi_j = 0$ **then**
      $LV = LV \cup \{j\}$
    end if
  end for
  if $LV \setminus SV = \varnothing$ `# Dual update:`
  then
    $\delta = \min\{j \in V \setminus LV\}$
    $\forall i \in SU$, $u_i = u_i + \delta$
    $\forall j \in LV$, $v_j = v_j - \delta$
    for $j \in V \setminus LV$ **do**
      $\pi_j = \pi_j - \delta$
      if $\pi_j = 0$ **then**
        $LV = LV \cup \{j\}$
      end if
    end for
  end if
  Chose $j \in LV \setminus SV$
  $SV = SV \cup \{j\}$
  if $row(j) = 0$ **then**
    $sink := j$
  else
    $i := row(j)$
  end if
end while
return $sink, pred, U, V$

Algorithm 3: Hungarian

Input : C
Output: row, x, phi, U, V, W
P, u, v=Initialize(C)
while $|AU| < 0$ **do**
  chose $k \in U \setminus AU$ `# Root of the tree`
  $sink$ =Augment(k) `# Build tree`
  $AU = AU \cup \{k\}$, $j = sink$
  repeat
    i = k
  until `# Update assignment`
  $:= pred_j$, $row(j) = i$, $h := \phi(i)$, $\phi(i) := j$, $j := h$
end while
return $sink, pred, U, V$

0.2.4 Numerical results

To obtain a discretization of a measure μ, it suffices to sample iid random variables of distribution μ [7, Section 8.4.1] The strong duality implies that having primal and dual satisfying (KKT), actually gives a certificate of optimality, which can be checked at the end of the algorithm. We implement the algorithm with Boolean arrays for all the sets, the code is available in the GitHub repository [6]. We test it on a Gaussian-distributed sample:

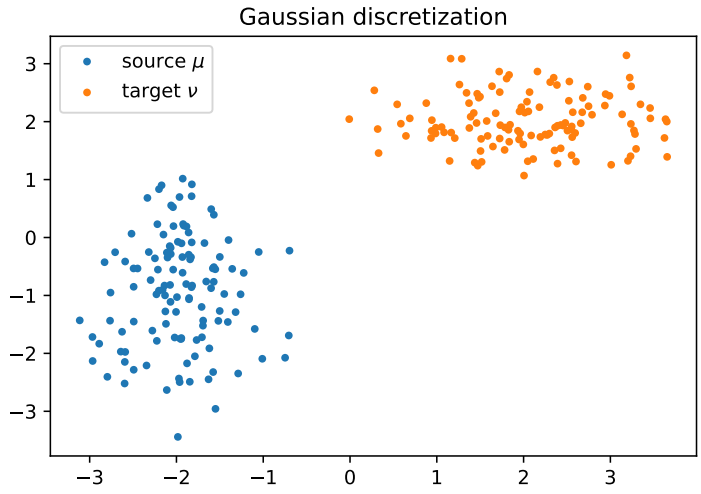

Figure 1: Discretization of two Gaussians on the plan

We use the analytic expression of optimal transport between Gaussian distribution to compare the true cost 25.4, to the cost obtained by the method, which is 25.24, so the relative error is of 0.6%. We display the geodesic at some selected points in Figure [10]. We also compute the evolution of the running time and the error with respect to the discretization in Figure [3]. We see that the order is not cubic but 1.77. This could be a consequence of the particular structure

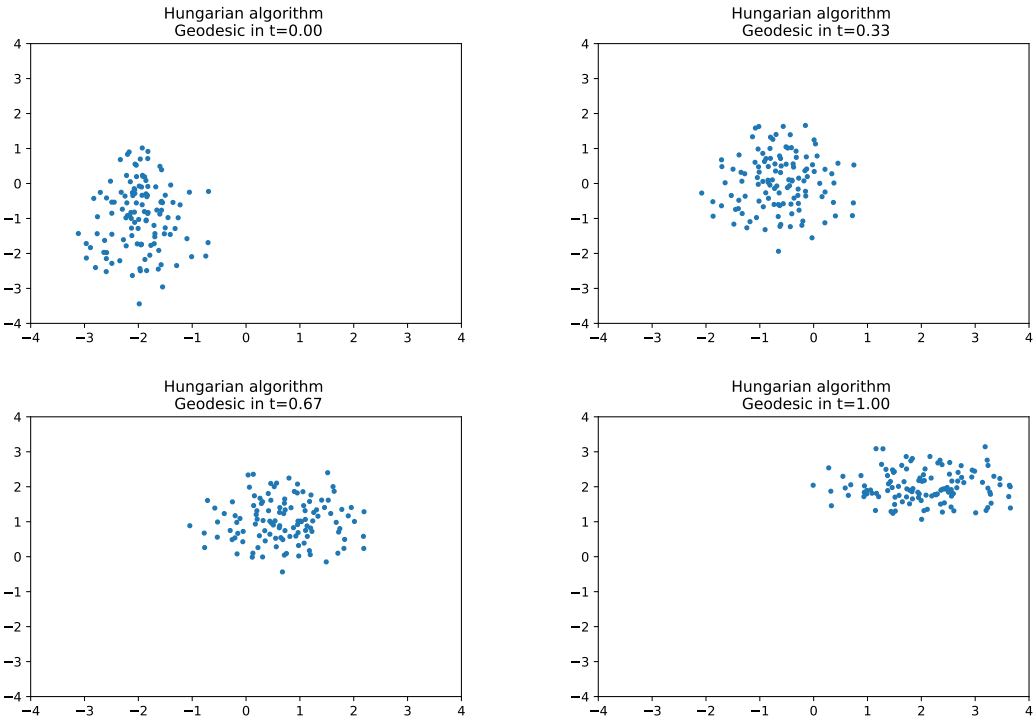

Figure 2: Hungarian algorithm: Geodesic of Gaussians

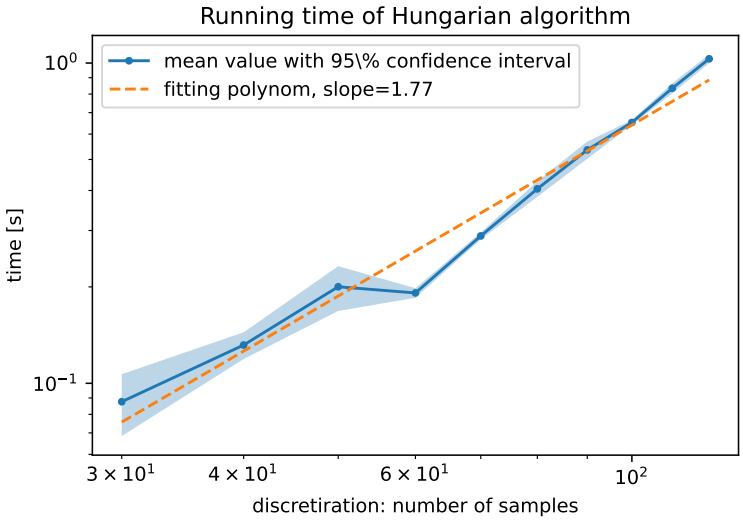

Figure 3: Hungarian algorithm: Performances

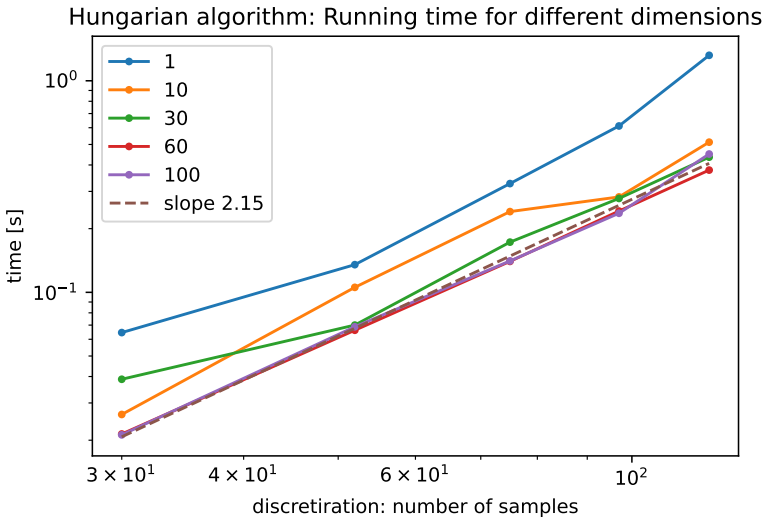

Figure 4: Hungarian algorithm: running time with respect to discretization

of the cost generated from a squared distance. To investigate this, we do the same computations for different values of dimension in Figure [4]. We see that the dimension seems to have an impact, on the value of the runtime, but not on the order. Let's plot the running time for the highest discretization and for different dimensions in Figure [5]. We see a significant effect, the running time seems to decrease as the dimension increase. As the dimension increase, the typical distances between objects increase, this could have in impact on the way the algorithm behaves, and the average-case complexity could be smaller than the worst-case complexity. We also show that the algorithm behaves perfectly with nonsmooth densities, such as uniform densities, see the discretization in Figure [6], and the geodesic in Figure [7].

As a matter of comparison with Figure [5], we show the result when the cost is chosen randomly in Figure [8]. We again get a order 1.40s smaller than the worst-case.

0.3 Dynamical formulation

0.3.1 Benamou Brenier formula

The idea of the fluid dynamic formulation is to consider the transport not as a direct link between the two measures, but as a continuous path of measures, such that the total cost along the path is minimized. These paths are geodesics in the Wasserstein metric:

$$\forall t \in [0,1],\ \rho_t \text{ is a probability measure and } W_2(\rho_s, \rho_t) = (s-t)W_2(\mu,\nu).$$

In the case where we know the optimal transport map T, the geodesic is simply the push-forward of the interpolation of the transport map and the identity:

$$\rho_t = T_{t\#}\mu \text{ with } T_t = (1-t)\,\mathrm{Id} + tT.$$

Now what we want to do is find a characterization of ρ and express the minimization problem with it. Suppose T_t is the flow of some vector field v_t:

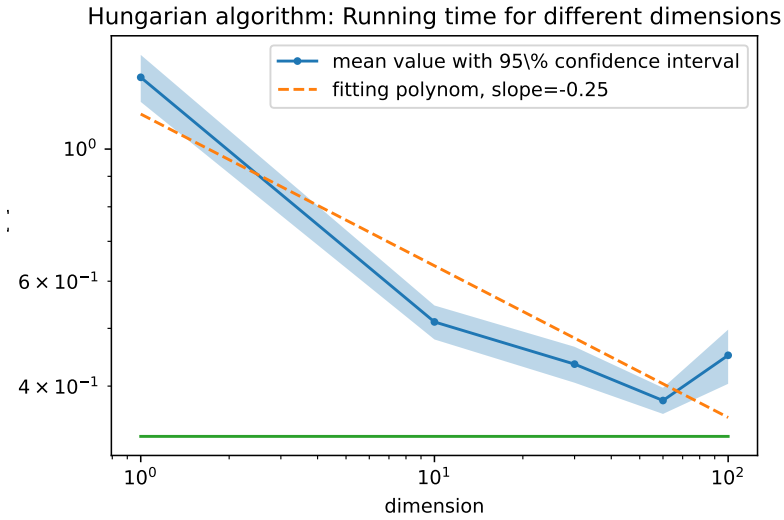

Figure 5: Hungarian algorithm: running time with respect to dimension

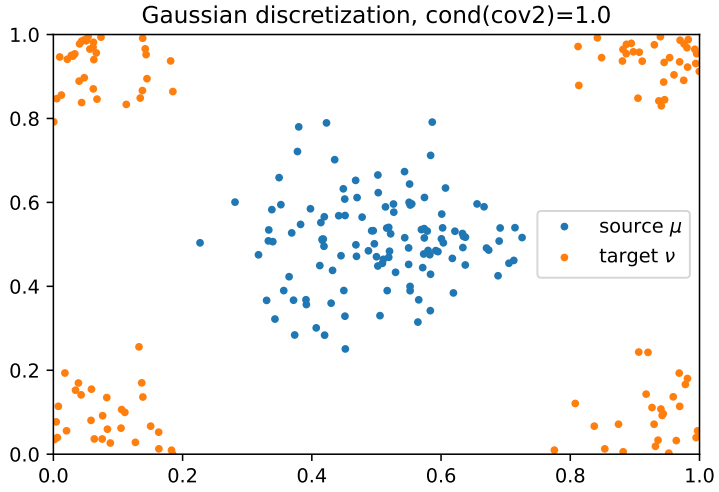

Figure 6: Hungarian algorithm: Uniform density on torus

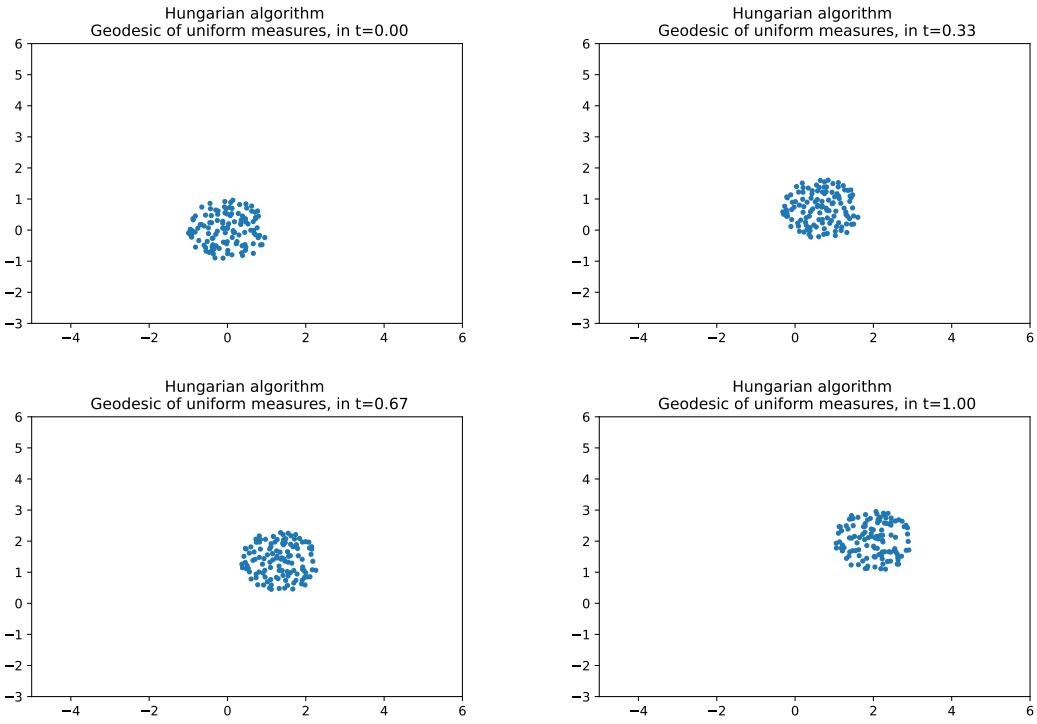

Figure 7: Hungarian algorithm: Geodesic for uniform density

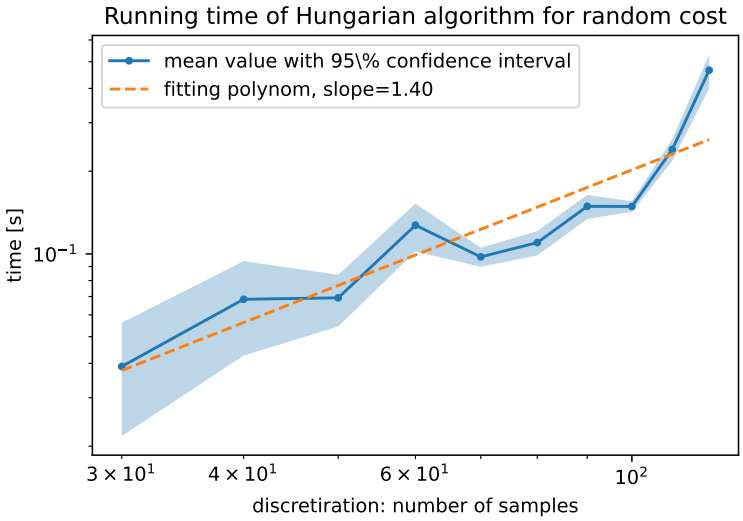

Figure 8: Hungarian algorithm: running time with respect to discretization for a random cost

$$\begin{cases} \partial_t T_t(x) = v_t(T_t(x)) \\ \rho_t = T_{t\#}\mu \end{cases}$$

Then we can derive a continuity equation:

Lemma 4 (Continuity equation)**.** *Let $(T_t)_{t\in[0,1]}$ be a locally Lipschitz family of diffeomorphisms with $T_0 = \mathrm{Id}$, and let $(v_t)_{t\in[0,1]}$ be the velocity fields associated to the trajectories T_t, i.e. vector field that satisfied the ODE just presented. Let μ be a source probability measure and $\rho_t := T_{t\#}\mu$. Then ρ is the unique solution to*

$$\partial_t \rho_t + \mathrm{div}(\rho_t v_t) = 0 \quad (\mathrm{div} \textit{ is in the weak sense})$$

in the $C([0,1], \mathcal{P}(\mathbb{R}^d))$ where the probabilities measure set is equipped with the weak topology. The weak formulation is

$$\forall \psi \in C_c^\infty(\mathbb{R}^d), \quad \partial_t \int \psi \,\mathrm{d}\rho_t = \int \langle \nabla\psi, v_t\rangle \,\mathrm{d}\rho_t$$

Proof. See Page 24 in the proof Section 0.5 □

For two probabilities measures μ, ν, we define the set $V(\mu,\nu)$ of couples (ρ, v) such that

- $\rho : [0,1] \to (\mathcal{P}_a c(\mathbb{R}^d), *\text{-weak})$ is a continuous curve of absolutely continuous measures in the $*$-weak topology[2].
- $\forall t \in [0,1], v_t \in L_2(\rho_t, \mathbb{R}^d)$ are vector fields
- $\cup_{t\in[0,1]} \mathrm{supp}(\rho_t)$ is bounded
- $\partial_t \rho_t + \mathrm{div}(\rho_t v_t) = 0$ in the weak sense
- $\rho_0 = \mu$, $\rho_1 = \nu$

Theorem 6 (Benamou-Brenier formula)**.**

$$W_2(\mu,\nu)^2 = \min\left\{ \int_0^1 \int \|v_t\|^2 \,\mathrm{d}\rho_t \,\mathrm{d}t \;\middle|\; \partial_t\rho_t + \mathrm{div}(\rho_t v_t) = 0,\ \rho_0 = \mu,\ \rho_1 = \nu \right\}$$

Proof. See Page 24 in the proof Section 0.5 □

0.3.2 Augmented Lagrangian optimization

The formulation in Theorem 6 induces an optimisation problem but the objective function is not jointly convex in (ρ, v), we make a change of variable to make it convex:

$$(\rho, v) \mapsto (\rho, \rho v) =: (\rho, m)$$

so that $\|v\|^2\rho = \frac{\|m\|^2}{\rho}$ when $\rho > 0$ and is 0 else. $k : (\rho, m) \mapsto \|m\|^2/\rho$ is now convex and can be expressed as a supremum of affine functions indexed on functions a, b that become dual variables:

Lemma 5.

$$k((\rho, m) = \sup_{(a,b)\in K} a\rho + \langle b, m\rangle$$

with $K = \{(a,b) : I \times \Omega \to \mathbb{R}\times\mathbb{R}^d \mid a + \frac{1}{2}\|b\|^2 \le 0\}$

[2] continuity of ρ_t in the *-weak topology if for all $\phi \in C_c(\mathbb{R}^n)$, $t \mapsto \int \phi \,\mathrm{d}\rho_t$ is continuous

Proof. See [8, Lemma 5.17]. □

Now, the continuity equation constraint becomes $0 = \partial_t \rho + \operatorname{div}(m) = \operatorname{div}_{t,x}(\rho, m)$, and using the boundary conditions we can derive a weak formulation:

Lemma 6. *Suppose that ρ and v are smooth densities and vector fields. Then the continuity equation and the boundary conditions are satisfied if and only if it is satisfied in the following sense:*

$$\forall \phi \in C_c^\infty([0,1], \bar{\Omega}),\ \int_0^1 \int_\Omega \rho \partial_t \phi + \langle \nabla \phi, m \rangle \,\mathrm{d}x\,\mathrm{d}t + G(\phi) = 0$$

with $G(\phi) = \int_\Omega \phi(0,x)\rho_0(x) - \phi(1,x)\rho_1(x)\,\mathrm{d}x$.

Proof. See [8, Proposition 4.3.]. □

As a result, we can use this expression to express the continuity condition in the objective function by introducing the test function ϕ as a dual variable, since as soon as the condition is not satisfied,

$$\sup_{\phi \in C_c^\infty(]0,1[,\bar{\Omega})} \left(\int_0^1 \int_\Omega \rho \partial_t \phi + \langle \nabla \phi, m \rangle \,\mathrm{d}x\,\mathrm{d}t + G(\phi) \right) = \infty \text{ if the condition is not satisfied}$$

.

Let us put everything together using Lemma 5 and Lemma 6 and rewrite it as a simple expression:

$$\begin{aligned}
W_2(\mu,\nu)^2 &= \inf \left\{ \int_0^1 \int \|v_t\|^2 \,\mathrm{d}\rho_t \,\mathrm{d}t \;\middle|\; \partial_t \rho_t + \operatorname{div}(\rho_t v_t) = 0,\ \rho_0 = \mu,\ \rho_1 = \nu \right\} \\
&= \inf \left\{ \int_0^1 \int_\Omega \sup_{(a,b)\in K} (a\rho + \langle b, m \rangle) \;\middle|\; \partial_t \rho_t + \operatorname{div}(\rho_t v_t) = 0,\ \rho_0 = \mu,\ \rho_1 = \nu \right\} \\
&= \inf_{\rho,m} \int_0^1 \int_\Omega \sup_{(a,b)\in K} (a\rho + \langle b, m \rangle) \,\mathrm{d}x\,\mathrm{d}t + \sup_\phi \left(- \int_0^1 \int_\Omega \rho \partial_t \phi + \langle \nabla, m \rangle \,\mathrm{d}x\,\mathrm{d}t - G(\phi) \right) \\
&= \inf_{\rho,m} \sup_{(a,b)\in K,\phi} \left(\int_0^1 \int_\Omega a\rho + \langle b, m \rangle - \rho \partial_t \phi - \langle \nabla \phi, m \rangle \,\mathrm{d}x\,\mathrm{d}t - G(\phi) \right) \\
&= \inf_{\rho,m} \sup_{(a,b)\in K,\phi} \left(\int_0^1 \int_\Omega (a - \partial_t \phi)\rho + \langle b - \nabla \phi, m \rangle \,\mathrm{d}x\,\mathrm{d}t - G(\phi) \right) \\
&= \inf_{\rho,m} \sup_{(a,b)\in K,\phi} \left(\int_0^1 \int_\Omega \left\langle (a,b) - \nabla_{x,t}\phi, (\rho, m) \right\rangle \mathrm{d}x\,\mathrm{d}t - G(\phi) \right) \\
&= \inf_{\rho,m} \sup_{(a,b)\in K,\phi} \left(\left\langle (a,b) - \nabla_{x,t}\phi, (\rho, m) \right\rangle_{L^2([0,1]\times\Omega)} - G(\phi) \right)
\end{aligned}$$

We obtain then a saddle problem, and we know that such problems arise when we formulate constrained optimization problems with Lagrange duality. The first term looks like a primal constraint and the second term could be the objective function. The only change to make here is inverting the role of the primal and dual variables and in order to have a minimization problem we reverse the direction of the extrema by taking the opposite value of the expression:

$$\begin{aligned}
-W_2(\mu,\nu)^2 &= - \inf_{\rho,m} \sup_{(a,b)\in K,\phi} \left(\left\langle (a,b) - \nabla_{x,t}\phi, (\rho, m) \right\rangle_{L^2([0,1]\times\Omega)} - G(\phi) \right) \\
&= \sup_{\rho,m} \inf_{(a,b)\in K,\phi} \left(G(\phi) + \left\langle (\rho, m), \nabla_{x,t}\phi - (a,b) \right\rangle_{L^2([0,1]\times\Omega)} \right)
\end{aligned}$$

Up to strong duality, this is just like if we had written the Lagrangian formulation of the minimization problem $\inf_{\nabla_{x,t}\phi \in K} G(\phi)$ with (ρ, m) as a dual variable. This motivate us to use the augmented Lagrangian method to try to solve the problem. We define $M = (\rho, m)$, $c = (a, b)$ and the augmented Lagrangian as

$$L_\tau(\phi, c, M) = G(\phi) + \langle M, \nabla_{x,t}\phi - c\rangle_{L^2([0,1]\times\Omega)} + \frac{\tau}{2}\|\nabla_{x,t}\phi - c\|^2_{L^2([0,1]\times\Omega)}$$

Algorithm 4: Augmented Lagrangian method (ALM)

Input : initial values ϕ^0, M^0, c^0
Output: A sequence of variables $(\phi^n, M^n, c^n)_n$
for $n = 0 \dots$ *and while convergence is not detected* **do**
  $(\phi^{n+1}, c^{n+1}) = \operatorname{argmin}_{(\phi,c):c\in K} L_\tau(\phi, c, M^n)$ (primal step)
  $M^{n+1} = M^{n+1} + \tau(\nabla_{x,t}\phi^{n+1} - c^{n+1})$ (dual step)
end for

The primal step is itself a constrained optimization problem, but it is constrained only for one of the two primal variables, so to be able to solve this sub-problem, we relax the optimization by doing two successive coordinate optimizations and we obtain a relaxed ALM:

Algorithm 5: Relaxed ALM (RALM)

Input : initial values ϕ^0, M^0, c^0
Output: A sequence of variables $(\phi^n, M^n, c^n)_n$
for $n = 0 \dots$ *and while convergence is not detected* **do**
  $\phi^{n+1} = \operatorname{argmin}_\phi L_\tau(\phi, c^n, M^n)$ (first primal step)
  $c^{n+1} = \operatorname{argmin}_{c\in K} L_\tau(\phi^{n+1}, c, M^n)$ (second primal step)
  $M^{n+1} = M^{n+1} + \tau(\nabla_{x,t}\phi^{n+1} - c^{n+1})$ (dual step)
end for

We detail the resolution of each of the two primal:

The First primal step can be solved by computing the first variation and setting it to zero. We obtain then the weak formulation of a Poisson equation with heterogeneous Neumann boundary conditions in time:

$$\begin{cases} \tau\Delta_{x,t}\phi = \operatorname{div}_{x,t}(\tau c^n - M^n) \\ \tau\partial_t\phi^n(0,.) = \mu - \rho^n(0,.) + \tau a^n(0,.) \\ \tau\partial_t\phi^n(1,.) = \nu - \rho^n(1,.) + \tau a^n(1,.) \end{cases}$$

The second primal step is solved by expanding the square and re-factorize the expression. Up to terms independent of c, we get

$$\inf_{c\in K} \|\nabla_{x,t}\phi^{n+1} + \frac{1}{\tau}M^n - c\|^2$$

which is minimal in the point-wise projection of $\nabla_{x,t}\phi^{n+1} + \frac{1}{\tau}M^n$ into the convex set K. The set

$$K = \{(a, b) : I \times \Omega \to \mathbb{R} \times \mathbb{R}^d \mid a + \frac{1}{2}\|b\|^2 \le 0\}$$

is generated by the rotation of the parabola $x + \frac{1}{2}y^2 \le 0$ around the x-axis and is hence convex. The projection is unique and computable by solving the normal equations.

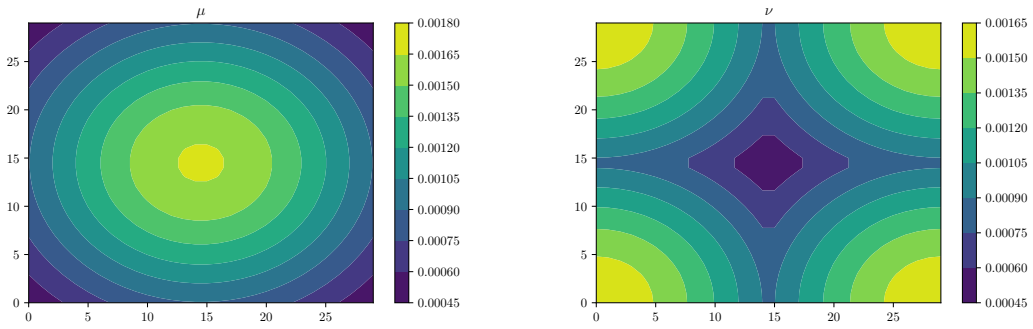

Figure 9: Initial and target Gaussian densities on a torus

0.3.3 Numerical results

To get rid of the space boundary conditions, we work on a torus, and we are interested in the 2D case. All the code is adapted for any dimension, except for the Poisson step which is only implemented for a 2D space (and hence a 3D space-time Poisson equation), the ND case works in a similar way and the implementation here can be generalized.

Let us detail the choices made to solve the primal steps:

The Poisson step (first primal step) is solved by finite differences on a regular grid, with a centered three points stencil for the second derivatives (second order). The Neumann boundary conditions are included by using ghost points and a centered two point stencil (second order). The stiffness matrix is the same at every step, so it is built one time only, and in sparse matrix mode.

To profit from the fact that the matrix never changes, we also compute and store a sparse LU factorization, by the approximate minimum degree column ordering method [3]. We can use then this factorization to solve the system at each step. The building of the factorization uses heuristics, but we are going to show numerically that in our case, the result is indeed sparse enough to reach almost linear complexity.

The asymptotic error is of order two with respect to the step size, and the complexity with respect to the total number of points. It is less than cubic since this is the complexity of the dense LU decomposition. Once the factorization is computed, solving the system is linear in the number of non-zero entries in the factorization. Hence if the factorization was really sparse, solving the system using a factorization would have linear complexity.

The projection step (second primal step) is done using the invariance of K under rotation. We reduce the problem to a 2D projection, and write the normal equations: they consist of finding the biggest real root of a cubic polynomial. This root is found using Cardano's method.

We test the implementation on two Gaussians, see their density in Figure [9]. The geodesic obtained is plotted in

We see a very smooth geodesic that divides the bump of the Gaussian into four bumps, that translate to the corners, which actually represent the same place. This is confirmed by the original result as in the paper [1] of Benamou-Brenier who proposed the method. We also

[3] By default method of the SciPy [11] function scipy.sparse.linalg.factorized

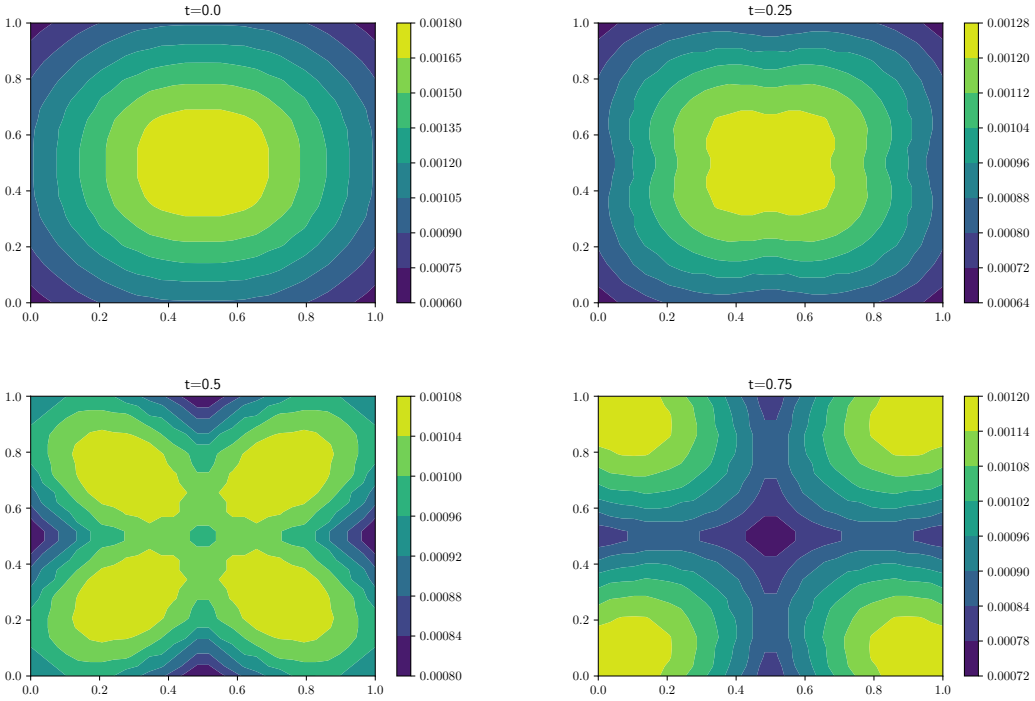

Figure 10: Benamou Brenier, Geodesic between Gaussian densities

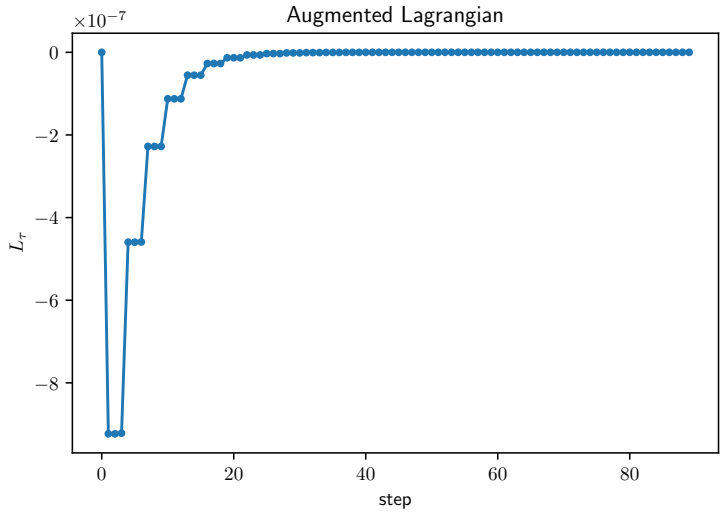

Figure 11: Benamou-Brenier: evolution of the augmented Lagrangian during the optimization

plot the evolution of the augmented Lagrangian in Figure [11]. We see that in around 30 steps, the curve stabilizes itself. The minimization step of the primal variables and the augmenting gradient step of the dual variable, tend to stop having effect, indicating that we are near a saddle point.

Let's study the running time of the method. We first plot the running time of the initialization in Figure [12], and see that the order is 2.26, so it is definitely better than the cubic order for dense matrices, however still not quadratic. But now we look at the running time per iteration, which we hoped to be almost linear. This is confirmed by Figure [13] that shows an order of 0.9. This result needs to be taken with caution, it might not be the actual complexity asymptotically. Although this doesn't take into account the initialization, this is very near to the $O(n\log(n))$ Fast Fourier Transform methods and the $O(n)$ multigrid methods [5]. Now how do these two complexities add up? Of course, if we add them up, we know that the complexity will be the biggest of the two, so 2.26. But we do it from a numerical perspective, to take into account the multiplicative constants that have a role for small numbers. We fix the number of iterations to 30 (like in the original paper), compute the total time, and plot with respect to the discretization. This is shown in Figure [14] looks like it has an order of 1.06. We lose indeed the linearity but for the typical values that we use, we definitely gained some advancement. To really compare numerically and absolutely, we are going to compare the actual times. Since the complexity of the iterations is better when we store the factorization, and the initialization is actually done one time only, we know that there exists a threshold, such that for enough iterations it is better to use the factorization. We compare this in Figure [15]. We clearly see the advantage: for around 30 iterations and discretization of $20^3 = 8000$, we are ten times faster.

0.4 Comparison of the two formulations

The two methods are fundamentally different.

The first one is discrete and uses very few structure. One can easily say that the shape and regularity of the density have no effect on how well the method will work. It can even deal with

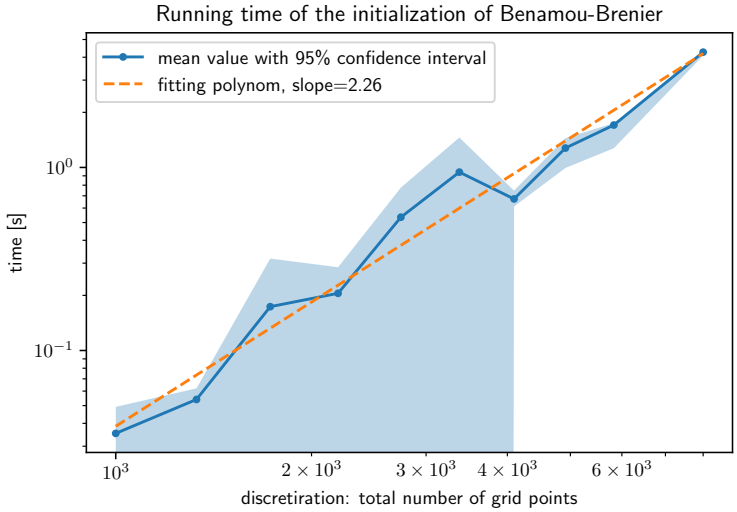

Figure 12: Benamou-Brenier: initialization time

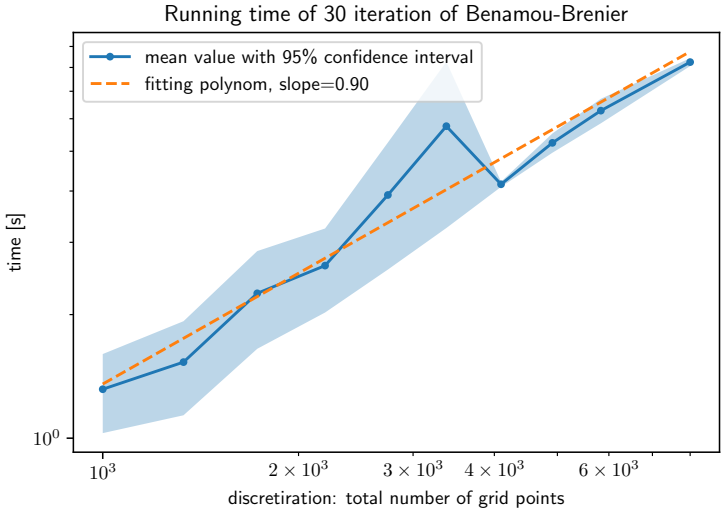

Figure 13: Benamou-Brenier: iteration time

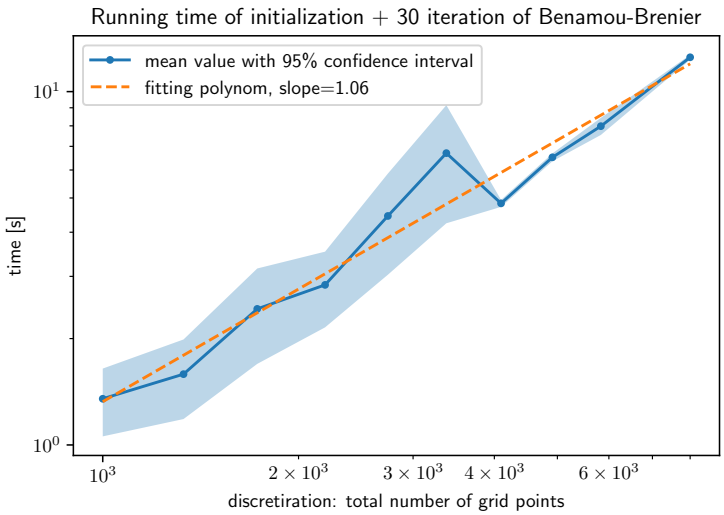

Figure 14: Benamou-Brenier: running time

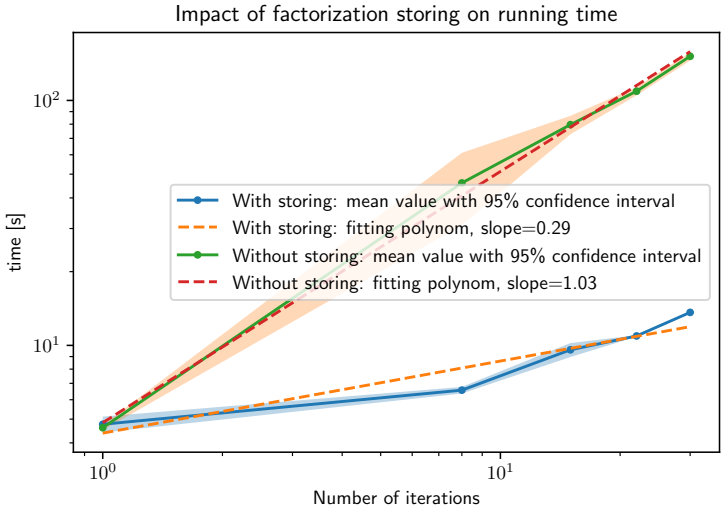

Figure 15: Benamou-Brenier: running time with and without factorization storage

mixed discrete-continuous measures. The fact that the Dirac measures positions are randomly sampled implies that we don't have to worry if the support of a measure is unbounded. Gaussian variables for example are perfectly handled. One could argue that since the number of samples is finite, the discretization cannot cover the whole support evenly, but the areas that the samples don't reach often are actually an area of small mass, and the more sample we have, the more area is covered. The Hungarian algorithm is actually an exact algorithm for finite supports.

In [7, Section 8.4.1], we are even given the bound

$$\mathbb{E}|W_2(\mu^{(N)}, \nu^{(N)}) - W_2(\mu, \nu)| \in O(n^{-1/d})$$

where $\mu^{(N)} = \frac{1}{N}\sum_{i_1}^{N} \delta_{X_i}$ is the empirical measure with X_i independent and identically μ-distributed (similar for ν). In other words, in the plan, the solution of the Hungarian algorithm is the exact optimal transport between the empirical measures, whose cost converges to the exact cost at average speed $1/\sqrt{n}$. Hence the average error of the Hungarian algorithm is $O(1/\sqrt{n})$.

It doesn't need any structure, but in counterpart, it cannot make a profit from any possible structure and regularity present. This is why the Benamou-Brenier method has the potential to outperform it for regular cases. We note that the way the Poisson equation is solved and the way it manages weak derivatives can impact the capacity of the algorithm to work on irregular measures. Moreover, the discretization of the measure with a mesh is not assured at all to be representative. Even if we have a density, the support could be totally disconnected from the mesh. If we just take the value of the density on the point, we don't have a guarantee. Instead, we should use the mass of the neighborhood cell around a mesh point (see Voronoil cells [8, Section 6.4.1]). We note that in a heavy area, a bad discretization will have a higher impact on the approximation, so here again, we see that the mesh shouldn't be regular but concentrated according to the density. For example, we could use random samples to determine the place where to put vertices for the mesh. Like so, it also deals with the choice of support for the mesh, and it doesn't lose computation capacity in light or not supported areas.

0.5 Proofs

Proof of Lemma 1. The condition for a map $T : \mathbb{R}^d \to \mathbb{R}^d$ to be a transport map is that for any Borel set $B \subset \mathbb{R}^d$, $\nu(B) = \mu(T^{-1}(B))$, so T needs only to be defined on the support of μ, $\text{supp}(\mu) = \{x_i\}_{i=1}^n$. The only choice for $T(x_i)$ is a certain y_j because

$$\nu(\{T(x_i)\}) = \mu(T^{-1}(\{T(x_i)\})) \geq \mu(\{x_i\}) = 1 > 0 \implies T(x_i) \in \{y_j\}_{j=1}^n.$$

Also, T is injective on the support because

$$\frac{1}{n} = \nu(\{y_j\}) = \mu(T^{-1}(\{y_j\})) \implies T^{-1}(\{y_j\}) = \{x_i\} \text{ for some i.}$$

hence, $T : \{x_i\}_{i=1}^n \to \{y_j\}_{j=1}^n$ is an injection between two sets of the same cardinalities, it is a bijection, and is associated with a permutation $\sigma \in S_n$ such that $T(x_i) = y_{\sigma(i)}$.

On the other way, it is clear that such maps are indeed transport maps, because for any $J \subset \{1, \dots, n\}$,

$$T(\{y_j\} j \in J) = |J|/n = \mu(\{T^{-1}(\{y_j\})\}_{j\in J}) = \mu(T^{-1}(\{y_j\}_{j\in J})).$$

□

Proof of Lemma 2. Let us compute the cost of transport associated with a transport map T of permutation σ:

$$\begin{aligned}
\int_{\mathbb{R}^d} \|x - T(x)\|^2 \,\mathrm{d}\mu(x) &= \sum_{i=1}^{n} \frac{1}{n} \int_{\mathbb{R}^d} \|x - T(x)\|^2 \,\mathrm{d}\delta_{x_i} \\
&= \sum_{i=1}^{n} \frac{1}{n} \|x_i - T(x_i)\|^2 \\
&= \sum_{i=1}^{n} C_{i\sigma(i)} && \text{with } C_{ij} = \tfrac{1}{n}\{x_i - y_j\}^2 \\
&= \sum_{i,j=1}^{n} P_{ij} C_{ij} && \text{with the permutation matrix } P_{ij} = \delta_{\sigma(i)j} \\
&= \langle P, C \rangle. && \text{with } \langle ., . \rangle \text{ the Frobenius scalar product}
\end{aligned}$$

The feasibility of P to represent a transport map is to be a permutation matrix, i.e.

$$\begin{cases} \sum_i P_{ij} = 1 & \forall j \in \{1, \dots, n\} \\ \sum_j P_{ij} = 1 & \forall i \in \{1, \dots, n\} \\ P_{ij} \in \{0, 1\} & \forall i, j \in \{1, \dots, n\} \end{cases} \quad \text{or} \quad \begin{cases} P\mathbb{1} = P^\top \mathbb{1} = \mathbb{1} & \text{with } \mathbb{1} = (1, \dots, 1)^\top \in \mathbb{R}^d \\ P \geq 0 & \text{point-wise} \\ P \in \mathbb{Z}^{d \times d} \end{cases}$$

This defines indeed an integer linear problem called linear sum assignment problem:

$$\min_{\substack{P_{ij} \in \{0,1\} \\ P\mathbb{1} = \mathbb{1} = P^\top \mathbb{1}}} \langle P, C \rangle$$

□

Proof of Lemma 3. We have here a linear program, so we use linear programming theory. Firstly, the identity matrix I_d is always a feasible variable and $\|P\|_1 = n$, so the feasible set is non-empty. It is also closed since inequality constraints are closed. It is finally bounded beacause the norm $\|P\|_{\text{Fro}} = n$ is bounded. As a result, the program admits a solution and is indeed a min.

Linear programming theory shows that the existence of solutions implies the existence of vertex solutions[9, Theorem 3.5]. Vertex solutions are solutions that cannot be written as a convex combination of other solutions. The vertices of the set Π_n of $n \times n$ doubly stochastic matrices are actually the permutations matrices, this is a consequence of Birkoff's Theorem [3, Theorem 2.18.] that says all doubly stochastic matrices are convex combinations of permutation matrices. This implies that vertices must be permutation matrices, and since permutation matrices are convex independent, they they are actually all vertices. To show convex independence, let us write zero as a positive convex combination of different permutation matrices. If we look entry-wise, we see that a positive linear combination of $0, 1$ is 0 if and only if all entries are 0. Since this is valid for all entries and all matrices are different we get a contradiction.

As a result, the scalar version of LSAP (1) always has integer solutions and in particular

$$\min_{\substack{P_{ij} \in \{0,1\} \\ P\mathbb{1} = \mathbb{1} = P^\top \mathbb{1}}} \langle P, C \rangle = \min_{\substack{P \geq 0 \\ P\mathbb{1} = \mathbb{1} = P^\top \mathbb{1}}} \langle P, C \rangle.$$

□

Sketch of proof of Lemma 4. Let $\psi \in C_c^\infty(\mathbb{R}^d)$, using the pushforward, we compute that

$$\begin{aligned}
\partial_t \int \psi \,\mathrm{d}\rho_t &= \partial_t \int \psi \circ T_t \,\mathrm{d}\mu \\
&= \int \partial_t(\psi \circ T_t) \,\mathrm{d}\mu \\
&= \int \langle \nabla\psi \circ T_t, \partial_t T_t \rangle \,\mathrm{d}\mu \\
&= \int \langle \nabla\psi \circ T_t, v_t \circ T_t \rangle \,\mathrm{d}\mu \\
&= \int \langle \nabla\psi, v_t \rangle \,\mathrm{d}T_{t\#}\mu \\
&= \int \langle \nabla\psi, v_t \rangle \,\mathrm{d}\rho_t
\end{aligned}$$

The unicity is not developed here, it follows from a duality argument, see [10, Theorem 5.34]. □

Sketch of proof of Theorem 6. See [10, Theorem 8.1] for complete proof. Let's begin by showing that if v is C^1, then $A(\rho, v) \geq W_2(\mu, \nu)^2$. For this, we can define T_t with the ODE

$$\begin{cases} \partial_t T_t(x) = v_t(T_t(x)) \\ T_0 = \mathrm{Id} \end{cases}$$

Since v is derivable and defined on the support of ρ which is bounded, it is Lipschitz continuous so the ODE is well defined and has a unique solution T such that curves T_t doesn't cross, so T_t is injective. By Lemma 4, $\rho_t = T_{t\#}\mu$ and then

$$\begin{aligned}
A(\rho, v) &= \int\int \|v_t(x)\|^2 \rho_t \,\mathrm{d}x \,\mathrm{d}t \\
&= \int\int \|v_t(T_t(x))\|^2 \rho_0 \,\mathrm{d}x \,\mathrm{d}t && \text{(by the push forward)} \\
&= \int\int \|\partial_t T_t(x)\|^2 \rho_0 \,\mathrm{d}t \,\mathrm{d}x && \text{(by Fubbini-Tonelli)} \\
&\geq \int \| \int \partial_t T_t(x) \,\mathrm{d}t\|^2 \rho_0 \,\mathrm{d}x && \text{(by Jensen inequality)} \\
&= \int \|T_1(x) - T_0(x)\|^2 \,\mathrm{d}\mu x && \text{(Fundamental theorem of analysis)} \\
&= \int \|T(x) - x\|^2 \,\mathrm{d}\mu x \\
&\geq W_2(\mu, \nu)^2.
\end{aligned}$$

We can show that the nonsmooth case can be reduced to the smooth case by a smoothing argument using convolutions. This has been made in [10, Theorem 8.1 p.239] but is not reproduced here for simplicity. We have then

$$W_2(\mu, \nu)^2 \leq \inf \left\{ \int_0^1 \int \|v_t\|^2 \,\mathrm{d}\rho_t \,\mathrm{d}t \;\middle|\; \partial_t \rho_t + \mathrm{div}(\rho_t v_t) = 0,\ \rho_0 = \mu,\ \rho_1 = \nu \right\},$$

and we exhibit a minimizer to get the equality and the fact that it is a min.

By Brenier Theorem 2, there exist a optimal transport map $T = \nabla\phi$, so we define the following interpolations

$$\begin{cases} T_t = (1-t)T + t\,\mathrm{Id} \\ \phi_t = (1-t)\phi + t\|\,\mathrm{Id}\,\|^2 \end{cases}$$

We get see that $T_t = \nabla\phi_t$ is the gradient of a convex map so again by Brenier Corollary **??** they are optimal maps, and $T_t^{-1} = \nabla\phi_t^*$ the gradient of the Legendre transform is the ρ_t-almost everywhere inverse of $\nabla\phi_t$([10, Proposition 2.4]). By evaluating the ODE in T_t^{-1} and using the definition of T_t, we can define

$$v_t = (\partial_t T_t) \circ T_t^{-1} = (T - \mathrm{Id}) \circ T_t^{-1}.$$

This v_t is bounded because $(T - \mathrm{Id}) \circ T_t^{-1}(\mathrm{supp}(\rho_t)) \subset (T - \mathrm{Id})(\mathrm{supp}(\mu)) \subset \mathrm{supp}(\nu) - \mathrm{supp}(\mu)$ is bounded. We define then $\rho_t = T_{t\#}\mu$, and by Lemma 4, (ρ, v) satisfy the continuity equation, and we compute

$$\begin{aligned} A(\rho, v) &= \int\int \|v_t(x)\|^2 \rho_t(x)\,\mathrm{d}x\,\mathrm{d}t \\ &= \int\int \|v_t(T_t(x))\|^2\,\mathrm{d}\mu(x)\,\mathrm{d}t \\ &= \int\int \|\partial_t T(x)\|^2\,\mathrm{d}\mu(x)\,\mathrm{d}t \\ &= \int\int \|T(x) - x\|^2\,\mathrm{d}\mu(x)\,\mathrm{d}t \\ &= \int \|T(x) - x\|^2\,\mathrm{d}\mu(x) \\ &= W_2(\mu,\nu)^2 \end{aligned}$$

which shows equality. □

List of Figures

1 Discretization of two Gaussians on the plan 9
2 Hungarian algorithm: Geodesic of Gaussians 10
3 Hungarian algorithm: Performances 10
4 Hungarian algorithm: running time with respect to discretization 11
5 Hungarian algorithm: running time with respect to dimension 12
6 Hungarian algorithm: Uniform density on torus 12
7 Hungarian algorithm: Geodesic for uniform density 13
8 Hungarian algorithm: running time with respect to discretization for a random cost 13
9 Initial and target Gaussian densities on a torus 17
10 Benamou Brenier, Geodesic between Gaussian densities 18
11 Benamou-Brenier: evolution of the augmented Lagrangian during the optimization 19
12 Benamou-Brenier: initialization time 20
13 Benamou-Brenier: iteration time 20
14 Benamou-Brenier: running time 21
15 Benamou-Brenier: running time with and without factorization storage 21

Bibliography

[1] Jean-David Benamou and Yann Brenier. "A computational fluid mechanics solution to the monge-kantorovich mass transfer problem". In: *Numerische Mathematik* 84 (2000), pp. 375–393.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. DOI: 10.1017/CBO9780511804441. URL: https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf.

[3] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems, Revised Reprint*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2012. ISBN: 9781611972221.

[4] Jonathan Eckstein. "Augmented Lagrangian and Alternating Direction Methods for Convex Optimization: A Tutorial and Some Illustrative Computational Results". In: 2012.

[5] David Bommes Mario Botsch and Leif Kobbelt. "Efficient Linear System Solvers for Mesh Processing". In: *Computer Graphics Group, RWTH Aachen Technical University* (). URL: https://www.graphics.rwth-aachen.de/media/papers/solvers1.pdf.

[6] Benoît Müller. URL: https://github.com/Benoit-Muller/Computational-Optimal-Transport.

[7] Gabriel Peyré and Marco Cuturi. *Computational Optimal Transport: with Applications to Data Sciences*. 2020. arXiv: 1803.00567 [stat.ML]. URL: https://arxiv.org/pdf/1803.00567.pdf.

[8] F. Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications. Springer International Publishing, 2015. ISBN: 9783319208282.

[9] Robert J. Vanderbei. *Linear Programming*. Vol. 196. Springer New York Heidelberg Dordrecht London, 2014. DOI: 10.1007/978-1-4614-7630-6. URL: https://dl.icdst.org/pdfs/files3/faa54c1f53965a11b03f9a13b023f9b2.pdf.

[10] C. Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN: 9780821833124.

[11] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.