COMMUN. MATH. SCI. © 2022 International Press
Vol. 20, No. 7, pp. 2053–2067

FAST COMMUNICATION

FAST SINKHORN I: AN $O(N)$ ALGORITHM FOR THE WASSERSTEIN-1 METRIC[*]

QICHEN LIAO[†], JING CHEN[‡], ZIHAO WANG[§], BO BAI[¶], SHI JIN[‖], AND HAO WU[**]

Abstract. The Wasserstein metric is broadly used in optimal transport for comparing two probabilistic distributions, with successful applications in various fields such as machine learning, signal processing, seismic inversion, etc. Nevertheless, the high computational complexity is an obstacle for its practical applications. The Sinkhorn algorithm, one of the main methods in computing the Wasserstein metric, solves an entropy regularized minimizing problem, which allows arbitrary approximations to the Wasserstein metric with $O(N^2)$ computational cost. However, higher accuracy of its numerical approximation requires more Sinkhorn iterations with repeated matrix-vector multiplications, which is still unaffordable. In this work, we propose an efficient implementation of the Sinkhorn algorithm to calculate the Wasserstein-1 metric with $O(N)$ computational cost, which achieves the optimal theoretical complexity. By utilizing the special structure of Sinkhorn's kernel, the repeated matrix-vector multiplications can be implemented with $O(N)$ times multiplications and additions, using the Qin Jiushao or Horner's method for efficient polynomial evaluation, leading to an efficient algorithm without losing accuracy. In addition, the log-domain stabilization technique, used to stabilize the iterative procedure, can also be applied in this algorithm. Our numerical experiments show that the newly developed algorithm is one to three orders of magnitude faster than the original Sinkhorn algorithm.

Keywords. Optimal transport; Wasserstein-1 metric; Sinkhorn algorithm; FS-1 algorithm; fast algorithm.

AMS subject classifications. 49M25; 65K10.

1. Introduction

The Wasserstein metric has been widely used in optimal transport for the global comparison between probabilistic distributions. It has been successfully used in various fields such as machine learning [17, 28], image processing [38], inverse problems [8, 14, 19, 33, 51], and density functional theory [7, 12, 21]. Many numerical methods have been developed, including the linear programming methods [26, 36, 50], combinatorial methods [42], solving Monge-Amphère equations [4, 5, 15, 16] and proximal splitting methods [11, 34]. In recent years, several approximation techniques in optimal transport for high-dimensional distributions have also been proposed approximately [31, 32].

One of the popular numerical techniques to compute the Wasserstein metric, is the the Sinkhorn algorithm [13, 45], which minimizes the entropy regularized problem. It provides the solution roughly in $O(N^2)$ operations with guaranteed convergence [29].

[*]Received: February 21, 2022; Accepted (in revised form): May 23, 2022. Communicated by Lexing Ying.

[†]Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China; Theory Lab, Central Research Institute, 2012 Labs, Huawei Technologies Co. Ltd., Hong Kong SAR, China (lqc20@mails.tsinghua.edu.cn).

[‡]School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore 639798 (jing.chen@ntu.edu.sg).

[§]Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong SAR, China (zwanggc@cse.ust.hk).

[¶]Theory Lab, Central Research Institute, 2012 Labs, Huawei Technologies Co. Ltd., Hong Kong SAR, China (baibo8@huawei.com).

[‖]Corresponding author. School of Mathematical Sciences, Institute of Natural Sciences, and MOE-LSC Shanghai Jiao Tong University, Shanghai 200240, China (shijin-m@sjtu.edu.cn).

[**]Corresponding author. Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China (hwu@tsinghua.edu.cn).

With the help of GPU acceleration, the efficiency of using the Sinkhorn algorithm to solve the optimal transport problem can be more significantly improved [40, 41]. For the Wasserstein-2 metric, the computation can be accelerated using the Gaussian convolution by approximating the geodesic distance-based kernel with the heat kernel [46]. Moreover, through statistical sampling, dimensional reduction, and other approximation methods, the complexity of the Sinkhorn algorithm can be reduced to $O(N\log N)$ [2,23] or even $O(N)$ [43]. An alternative way is to define the Wasserstein metric on the finite tree space, and the computational complexity could be $O(N\log N)$ [24, 47]. Because of these progresses, the Sinkhorn algorithm has been widely used in practical problems. However, those techniques provide just approximations of the transportation cost, rather than the precise computation of the original optimization problem. Therefore, it is still of great interest to develop fast and accurate Sinkhorn-type algorithms for solving large-scale optimal transport problems.

In recent years, some fast algorithms for accurately solving the optimal transport problem have been developed. For example, through multi-level grids, the computation complexity can be reduced to $O(N^{1.5}\log N)$ [30]. For the Wasserstein-1 distance, the equivalent Benamou-Brenier form has a very special structure, which is very common in compressive sensing, and thus can be efficiently solved with $O(N)$ algorithm complexity [25]. Another well-known conclusion is that for the one-dimensional quadratic Wasserstein metric, the complexity of the sorting-based algorithm is only $O(N\log N)$ [37].

In this work, by observing the special structure of the kernel matrix in the Sinkhorn algorithm to solve the Wasserstein-1 metric on uniform mesh, we propose a novel matrix-vector multiplication based on dynamic programming [22]. During each iteration step, it involves only a forward and backward recursive sweeping process, as in Qin Jiushao's (or Horners', although it appeared several centuries later) method for polynomial evaluations [20, 35], which reduces the computational cost from $O(N^2)$ to $O(N)$ in each step, thus achieving the optimal theoretical complexity. In addition, the log-stabilization [9], an important technique to improve the numerical stability of the Sinkhorn algorithm, can be implemented in this strategy with the same stability property. In this paper we abbreviate this method as FS-1.

The rest of the paper is organized as follows. In Section 2, the basics of the Wasserstein-1 metric and the Sinkhorn algorithm are briefly introduced. After showing the elaborate structure of the obtained kernel matrix, we use it to develop the FS-1 algorithm in Section 3. We will also analyze the stability and integrate the log-stabilization technique into our FS-1 to improve the numerical stability. In Section 4, the FS-1 is generalized to higher dimension. We will provide numerical experiments to illustrate the huge efficiency advantage of our FS-1 algorithm in Section 5. Finally, we conclude the paper in Section 6.

2. The Wasserstein-1 metric and the Sinkhorn algorithm

Consider two probabilistic density functions $u(x)$ and $v(x)$ on a domain $\Omega\subset\mathbb{R}$, the Kantorovich's formulation of Wasserstein-1 metric is defined as [48]:

$$\begin{aligned} W(u,v) &= \inf_{\gamma(x,y)\in\Gamma}\int_{\Omega\times\Omega} d(x,y)\gamma(x,y)\mathrm{d}x\mathrm{d}y, \\ \Gamma &= \left\{\gamma(x,y)\,\middle|\,\int_\Omega \gamma(x,y)\mathrm{d}y = u(x), \int_\Omega \gamma(x,y)\mathrm{d}x = v(y)\right\}, \end{aligned} \tag{2.1}$$

where $d(x,y)=|x-y|$. The Sinkhorn algorithm proposed in [13, 45] introduces an entropy term and solves the regularized problem:

$$W_\varepsilon(u,v) = \inf_{\gamma(x,y)\in\Gamma} \int_{\Omega\times\Omega} |x-y|\gamma(x,y) + \varepsilon\gamma(x,y)\ln\big(\gamma(x,y)\big)\mathrm{d}x\mathrm{d}y. \tag{2.2}$$

As for numerical realization, we consider two discretized probabilistic distributions

$$\boldsymbol{u} = (u_1, u_2, \cdots u_N), \quad \boldsymbol{v} = (v_1, v_2, \cdots, v_N),$$

on a uniform mesh grid with a grid spacing of h. Then the entropy regularized minimizing problem (2.2) can be discretized as

$$W_\varepsilon(\boldsymbol{u},\boldsymbol{v}) = \inf_{\gamma_{ij}} \sum_{i=1}^{N}\sum_{j=1}^{N} \big(\gamma_{ij}\,|i-j|\,h + \varepsilon\gamma_{ij}\ln\big(\gamma_{ij}\big)\big), \tag{2.3}$$

and γ_{ij} satisfies

$$\sum_{j=1}^{N}\gamma_{ij} = u_i, \quad \sum_{i=1}^{N}\gamma_{ij} = v_j, \quad \gamma_{ij} \geq 0.$$

The Lagrangian of the above equations writes

$$L(\boldsymbol{\gamma},\boldsymbol{\alpha},\boldsymbol{\beta}) = \sum_{i=1}^{N}\sum_{j=1}^{N}(\gamma_{ij}\,|i-j|\,h + \varepsilon\gamma_{ij}\ln(\gamma_{ij})) + \sum_{i=1}^{N}\alpha_i(\sum_{j=1}^{N}\gamma_{ij} - u_i) + \sum_{j=1}^{N}\beta_j(\sum_{i=1}^{N}\gamma_{ij} - v_j).$$

Taking the derivative of the Lagrangian with respect of γ_{ij} directly leads to

$$\gamma_{ij} = e^{-\frac{1}{2}-\frac{1}{\varepsilon}\alpha_i} K_{ij} e^{-\frac{1}{2}-\frac{1}{\varepsilon}\beta_j}, \text{ where } K_{ij} = e^{-|i-j|h/\varepsilon}.$$

To avoid ε in the denominator, setting $\phi_i = e^{-\frac{1}{2}-\frac{1}{\varepsilon}\alpha_i}$ and $\psi_j = e^{-\frac{1}{2}-\frac{1}{\varepsilon}\beta_j}$, we obtain

$$\phi_i\sum_{j=1}^{N}K_{ij}\psi_j = u_i, \quad \psi_j\sum_{i=1}^{N}K_{ij}\phi_i = v_j, \quad \forall i,j = 1,2,\cdots,N.$$

Since the entries in $K = (K_{ij})$ are strictly positive, Sinkhorn's iteration [45] can be applied to iteratively update vectors $\boldsymbol{\phi} = (\phi_i)$ and $\boldsymbol{\psi} = (\psi_j)$ by pointwise computation:

$$\boldsymbol{\psi}^{(\ell+1)} = \boldsymbol{v} \oslash (K^T\boldsymbol{\phi}^{(\ell)}), \quad \boldsymbol{\phi}^{(\ell+1)} = \boldsymbol{u} \oslash (K\boldsymbol{\psi}^{(\ell+1)}), \tag{2.4}$$

in which the notion $\oslash$ represents pointwise division and ℓ denotes the iterative steps.

Remark 2.1. In [9], a log-domain stabilization technique is proposed to reduce the numerical instability caused by the small parameter ε. The idea is that when the infinite norms of $\boldsymbol{\phi}$ or $\boldsymbol{\psi}$ exceed a given threshold τ, these two vectors will be normalized with the excessive part 'absorbed' in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \varepsilon\ln(\boldsymbol{\phi}^{(\ell)}), \quad \boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \varepsilon\ln(\boldsymbol{\psi}^{(\ell)}), \quad \boldsymbol{\phi}^{(\ell)} \leftarrow \mathbf{1}_N, \quad \boldsymbol{\psi}^{(\ell)} \leftarrow \mathbf{1}_N.$$

Correspondingly, the matrix K needs to be rescaled as $\mathcal{K} \leftarrow diag(e^{\boldsymbol{\alpha}/\varepsilon}) \times K \times diag(e^{\boldsymbol{\beta}/\varepsilon})$.

3. The FS-1 Algorithm

The key to the Sinkhorn algorithm is to iteratively update $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ through Equation (2.4). By introducing the notation $\lambda = e^{-h/\varepsilon}$, the matrix multiplication vector operation is written as

$$K\boldsymbol{\psi}^{(\ell)} = \begin{pmatrix} \psi_1^{(\ell)} & + & \lambda\psi_2^{(\ell)} & + & \lambda^2\psi_3^{(\ell)} & \cdots + \lambda^{N-1}\psi_N^{(\ell)} \\ \lambda\psi_1^{(\ell)} & + & \psi_2^{(\ell)} & + & \lambda\psi_3^{(\ell)} & \cdots + \lambda^{N-2}\psi_N^{(\ell)} \\ \lambda^2\psi_1^{(\ell)} & + & \lambda\psi_2^{(\ell)} & + & \psi_3^{(\ell)} & \cdots + \lambda^{N-3}\psi_N^{(\ell)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \vdots \quad \vdots \\ \lambda^{N-1}\psi_1^{(\ell)} & + & \lambda^{N-2}\psi_2^{(\ell)} & + & \lambda^{N-3}\psi_3^{(\ell)} & \cdots + \quad \psi_N^{(\ell)} \end{pmatrix}. \tag{3.1}$$

We separate the summation of row k to the lower triangular part p_k and the strictly upper triangular part q_k. Then updating $\boldsymbol{\phi}^{(\ell+1)}$ is formulated as

$$\phi_k^{(\ell+1)} = u_k/(p_k+q_k), \quad p_k = \sum_{i=1}^{k} \psi_i^{(\ell)}\lambda^{k-i}, \quad q_k = \sum_{i=k+1}^{N} \psi_i^{(\ell)}\lambda^{i-k}, \quad k=1,2,\cdots,N.$$

Instead of directly calculating p_k and q_k, we use the recursive computation given by

$$\begin{aligned} &p_1 = \psi_1^{(\ell)}, \quad p_{k+1} = \lambda p_k + \psi_{k+1}^{(\ell)}, \quad k=1,2,\cdots,N-1, \\ &q_N = 0, \quad q_k = \lambda(q_{k+1} + \psi_{k+1}^{(\ell)}), \quad k=N-1,N-2,\cdots,1. \end{aligned} \tag{3.2}$$

Thus we develop the FS-1 algorithm with linear computational complexity, which only takes $2(N-1)$ times additions and multiplications for the matrix multiplication operation. The pseudo-code is presented in Algorithm 1.

Algorithm 1 FS-1 Algorithm

```
Input: $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^N$; itr_max $\in \mathbb{N}^+$; $h, \varepsilon$, tol $\in \mathbb{R}$
Output: $W_\varepsilon(\boldsymbol{u},\boldsymbol{v})$
1:  $\lambda \leftarrow e^{-h/\varepsilon}$; $\boldsymbol{\phi},\boldsymbol{\psi} \leftarrow \frac{1}{N}\mathbf{1}_N$; $\boldsymbol{p},\boldsymbol{r},\boldsymbol{q},\boldsymbol{s} \leftarrow \mathbf{0}_N$; $\ell \leftarrow 0$
2:  while ($\ell <$ itr_max) and $\left(\sum_{i=1}^N \left|\psi_i(\sum_{j=1}^N \phi_j \lambda^{|i-j|}) - v_i\right| > \text{tol}\right)$ do
3:      $r_1 \leftarrow \phi_1$, $s_N \leftarrow 0$
4:      for $i = 1 : N-1$ do
5:          $r_{i+1} \leftarrow \lambda r_i + \phi_{i+1}$
6:          $s_{N-i} \leftarrow \lambda(s_{N-i+1} + \phi_{N-i+1})$
7:      $\boldsymbol{\psi} \leftarrow \boldsymbol{v} \oslash (\boldsymbol{r}+\boldsymbol{s})$
8:      $p_1 \leftarrow \psi_1$, $q_N \leftarrow 0$
9:      for $i = 1 : N-1$ do
10:         $p_{i+1} \leftarrow \lambda p_i + \psi_{i+1}$
11:         $q_{N-i} \leftarrow \lambda(q_{N-i+1} + \psi_{N-i+1})$
12:     $\boldsymbol{\phi} \leftarrow \boldsymbol{u} \oslash (\boldsymbol{p}+\boldsymbol{q})$
13:     $\ell \leftarrow \ell + 1$
    return $\sum_{i,j=1}^N \phi_i \psi_j \lambda^{|i-j|} |i-j| h$
```

Remark 3.1. The recursion (3.2) is actually the Qin Jiushao [35] or Horner method [20] for efficient polynomial evaluation:

$$f(x) = \sum_{k=1}^{N} \psi_k^{(\ell)} \lambda^{N-k} = \psi_N^{(\ell)} + \lambda\left(\psi_{N-1}^{(\ell)} + \lambda\left(\psi_{N-2}^{(\ell)} + \cdots + \lambda\left(\psi_2^{(\ell)} + \lambda\psi_1^{(\ell)}\right)\cdots\right)\right).$$

Thus, we can recursively obtain

$$p_1 = \psi_1^{(\ell)}, \quad p_2 = \psi_2^{(\ell)} + \lambda\psi_1^{(\ell)}, \quad p_3 = \psi_3^{(\ell)} + \lambda\psi_2^{(\ell)} + \lambda^2\psi_1^{(\ell)}, \quad \cdots,$$
$$p_N = \psi_N^{(\ell)} + \lambda\psi_{N-1}^{(\ell)} + \cdots + \lambda^{N-1}\psi_1^{(\ell)},$$

with an overall computational cost of $O(N)$.

It is well-known that the Sinkhorn algorithm has stability issues [9] due to the division and the multiplication of the small parameter λ^k. Next, we need to discuss the stability of the FS-1 algorithm to ensure that its stability is not worse than the Sinkhorn algorithm.

Consider the matrix multiplication in (3.1),

$$S^\alpha = K\boldsymbol{\psi}^\alpha, \quad S^\alpha = (S_1^\alpha, S_2^\alpha, \cdots, S_N^\alpha)^T, \quad \boldsymbol{\psi}^\alpha = (\psi_1^\alpha, \psi_2^\alpha, \cdots, \psi_N^\alpha)^T, \quad \alpha = 1,2.$$

Assume that

$$|\psi_k^1 - \psi_k^2| \le \delta, \quad k = 1,2,\cdots,N,$$

thus

$$\begin{aligned}|S_k^1 - S_k^2| &= |\lambda^{k-1}(\psi_1^1 - \psi_1^2) + \lambda^{k-2}(\psi_2^1 - \psi_2^2) + \cdots + (\psi_k^1 - \psi_k^2) + \cdots + \lambda^{N-k}(\psi_N^1 - \psi_N^2)|,\\ &\le |\lambda^{k-1} + \lambda^{k-2} + \cdots + \lambda + 1 + \lambda + \cdots + \lambda^{N-k}|\,\delta = \left(\frac{2-(\lambda^k + \lambda^{N-k+1})}{1-\lambda} - 1\right)\delta.\end{aligned}$$

On the other hand, consider the successive computation in (3.2), an easy induction gives

$$|p_k^1 - p_k^2| \le \lambda|p_{k-1}^1 - p_{k-1}^2| + |\psi_k^1 - \psi_k^2| \le \cdots \le (1 + \lambda + \cdots \lambda^{k-1})\,\delta = \frac{1-\lambda^k}{1-\lambda}\delta,$$

and

$$|q_k^1 - q_k^2| \le \lambda|q_{k+1}^1 - q_{k+1}^2| + |\psi_{k+1}^1 - \psi_{k+1}^2| \le \cdots \le (\lambda + \cdots + \lambda^{N-k-1})\,\delta \le \left(\frac{1-\lambda^{N-k}}{1-\lambda} - 1\right)\delta.$$

Thus, we have

$$|(p_k^1 + q_k^1) - (p_k^2 + q_k^2)| \le \left(\frac{2-(\lambda^k + \lambda^{N-k+1})}{1-\lambda} - 1\right)\delta,$$

that is, the FS-1 algorithm and the Sinkhorn algorithm have the same stability.

REMARK 3.2. Similarly, the log-domain stabilization [9] technique can also be aggregated into the FS-1 algorithm. First, we need to 'absorb' the excessive part of vectors $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ into $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \varepsilon\ln(\boldsymbol{\phi}^{(\ell)}), \quad \boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \varepsilon\ln(\boldsymbol{\psi}^{(\ell)}), \quad \boldsymbol{\phi}^{(l)} \leftarrow \mathbf{1}_N, \quad \boldsymbol{\psi}^{(l)} \leftarrow \mathbf{1}_N.$$

Next, we have to replace lines 5-6 and 10-11 in the Algorithm 1 as

$$\begin{aligned}5:&\quad r_{i+1}^{(\ell)} = \lambda e^{(\beta_{i+1}-\beta_i)/\varepsilon} r_i^{(\ell)} + e^{(\alpha_{i+1}+\beta_{i+1})/\varepsilon}\phi_{i+1}^{(\ell)},\\ 6:&\quad s_{N-i}^{(\ell)} = \lambda e^{(\beta_{N-i}-\beta_{N-i+1})/\varepsilon}\left(s_{N-i+1}^{(\ell)} + e^{(\alpha_{N-i+1}+\beta_{N-i+1})/\varepsilon}\phi_{N-i+1}^{(\ell)}\right),\end{aligned}$$

```
10:  $p_{i+1}^{(\ell)} = \lambda e^{(\alpha_{i+1}-\alpha_i)/\varepsilon} p_i^{(\ell)} + e^{(\alpha_{i+1}+\beta_{i+1})/\varepsilon} \psi_{i+1}^{(\ell+1)}$,
11:  $q_{N-i}^{(\ell)} = \lambda e^{(\alpha_{N-i}-\alpha_{N-i+1})/\varepsilon} \left( q_{N-i+1}^{(\ell)} + e^{(\alpha_{N-i+1}+\beta_{N-i+1})/\varepsilon} \psi_{N-i+1}^{(\ell+1)} \right)$.
```

This leads to the stabilized FS-1 algorithm with $O(N)$ complexity.

REMARK 3.3. In practice, the direct calculation of $\lambda^m = e^{-mh/\varepsilon}$ and the multiplication of λ^m may be troublesome since λ^m could be very small. However, the FS-1 algorithm gets around this problem by stepwise multiplication of λ.

4. Extension to high dimension

In this section, we illustrate how the FS-1 algorithm generalizes to higher dimensions using the two-dimensional case as an example. First, consider two discretized probabilistic distributions

$$\boldsymbol{u} = (u_{11}, u_{21}, \cdots, u_{N1}, u_{12}, \cdots, u_{i_1 j_1}, \cdots, u_{NM}),$$
$$\boldsymbol{v} = (v_{11}, v_{21}, \cdots, v_{N1}, v_{12}, \cdots, v_{i_2 j_2}, \cdots, v_{NM}),$$

on a uniform 2D mesh of size $N \times M$ with a vertical spacing of h_1 and a horizontal spacing of h_2, the entropy regularized 2D Wasserstein-1 metric can be discretized as the optimal value of the following minimizing problem:

$$W_\varepsilon(\boldsymbol{u}, \boldsymbol{v}) = \inf_{\Gamma_{i_1 j_1 i_2 j_2}} \sum_{i_1, i_2 = 1}^{N} \sum_{j_1, j_2 = 1}^{M} \left(\Gamma_{i_1 j_1 i_2 j_2} (|i_1 - i_2| h_1 + |j_1 - j_2| h_2) + \varepsilon \Gamma_{i_1 j_1 i_2 j_2} \ln \left(\Gamma_{i_1 j_1 i_2 j_2} \right) \right), \tag{4.1}$$

where $\Gamma_{i_1 j_1 i_2 j_2}$ satisfies

$$\sum_{i_2=1}^{N} \sum_{j_2=1}^{M} \Gamma_{i_1 j_1 i_2 j_2} = u_{i_1 j_1}, \quad \sum_{i_1=1}^{N} \sum_{j_1=1}^{M} \Gamma_{i_1 j_1 i_2 j_2} = v_{i_2 j_2}, \quad \Gamma_{i_1 j_1 i_2 j_2} \geq 0.$$

Same as in the 1D case, the problem (4.1) can be solved by the Sinkhorn iteration. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are flattened into 1D vectors in column-major order, the corresponding kernel matrix is written as

$$K = \left(\begin{array}{c|c|c|c|c} K_0 & \lambda_2 K_0 & \lambda_2^2 K_0 & \cdots & \lambda_2^{M-1} K_0 \\ \hline \lambda_2 K_0 & K_0 & \lambda_2 K_0 & \cdots & \lambda_2^{M-2} K_0 \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline \lambda_2^{M-1} K_0 & \lambda_2^{M-2} K_0 & \lambda_2^{M-3} K_0 & \cdots & K_0 \end{array} \right),$$

where the sub-matrix

$$K_0 = \begin{pmatrix} 1 & \lambda_1 & \cdots & \lambda_1^{N-1} \\ \lambda_1 & 1 & \cdots & \lambda_1^{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^{N-1} & \lambda_1^{N-2} & \cdots & 1 \end{pmatrix},$$

and

$$\lambda_1 = e^{-h_1/\varepsilon}, \quad \lambda_2 = e^{-h_2/\varepsilon}.$$

Obviously, the cost of direct matrix-vector multiplication in Sinkhorn is $O(N^2M^2)$. By using the FS-1's trick twice, we expect the computational cost can be significantly reduced, the key idea is as follows. Let

$$\boldsymbol{\psi}_i^{(\ell)} = \left(\psi_{1i}^{(\ell)}, \psi_{2i}^{(\ell)}, \cdots, \psi_{Ni}^{(\ell)}\right), \quad i = 1,2,\cdots,M,$$

the matrix-vector multiplication $K\boldsymbol{\psi}^{(\ell)}$ is written as

$$K\boldsymbol{\psi}^{(\ell)} = \begin{pmatrix} K_0\boldsymbol{\psi}_1^{(\ell)} & + & \lambda_2 K_0\boldsymbol{\psi}_2^{(\ell)} & + & \lambda_2^2 K_0\boldsymbol{\psi}_3^{(\ell)} & \cdots + & \lambda_2^{M-1}K_0\boldsymbol{\psi}_M^{(\ell)} \\ \lambda_2 K_0\boldsymbol{\psi}_1^{(\ell)} & + & K_0\boldsymbol{\psi}_2^{(\ell)} & + & \lambda_2 K_0\boldsymbol{\psi}_3^{(\ell)} & \cdots + & \lambda_2^{M-2}K_0\boldsymbol{\psi}_M^{(\ell)} \\ \lambda_2^2 K_0\boldsymbol{\psi}_1^{(\ell)} & + & \lambda_2 K_0\boldsymbol{\psi}_2^{(\ell)} & + & K_0\boldsymbol{\psi}_3^{(\ell)} & \cdots + & \lambda_2^{M-3}K_0\boldsymbol{\psi}_M^{(\ell)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \vdots & \vdots \\ \lambda_2^{M-1}K_0\boldsymbol{\psi}_1^{(\ell)} & + & \lambda_2^{M-2}K_0\boldsymbol{\psi}_2^{(\ell)} & + & \lambda_2^{M-3}K_0\boldsymbol{\psi}_3^{(\ell)} & \cdots + & K_0\boldsymbol{\psi}_M^{(\ell)} \end{pmatrix}. \tag{4.2}$$

We separate the summation of row k to the lower triangular part $\boldsymbol{p}_k$ and the strictly upper triangular part $\boldsymbol{q}_k$. Then updating $\boldsymbol{\phi}^{(\ell+1)}$ is formulated as

$$\boldsymbol{\phi}_k^{(\ell+1)} = \boldsymbol{u}_k \oslash (\boldsymbol{p}_k + \boldsymbol{q}_k), \quad \boldsymbol{p}_k = \sum_{i=1}^{k} K_0\boldsymbol{\psi}_i^{(\ell)}\lambda_2^{k-i}, \quad \boldsymbol{q}_k = \sum_{i=k+1}^{M} K_0\boldsymbol{\psi}_i^{(\ell)}\lambda_2^{i-k}, \quad k = 1,\cdots,M.$$

Instead of directly calculating $\boldsymbol{p}_k$ and $\boldsymbol{q}_k$, a successive computation is used

$$\begin{aligned} &\boldsymbol{p}_1 = K_0\boldsymbol{\psi}_1^{(\ell)}, \quad \boldsymbol{p}_{k+1} = \lambda_2\boldsymbol{p}_k + K_0\boldsymbol{\psi}_{k+1}^{(\ell)}, \quad k = 1,2,\cdots,M-1, \\ &\boldsymbol{q}_M = \boldsymbol{0}_N, \quad \boldsymbol{q}_k = \lambda_2(\boldsymbol{q}_{k+1} + K_0\boldsymbol{\psi}_{k+1}^{(\ell)}), \quad k = M-1, M-2,\cdots,1. \end{aligned} \tag{4.3}$$

The computation of $K_0\boldsymbol{\psi}_k^{(\ell)}$ can be carried out by using recursion (3.2) in $O(N)$ complexity. Thus, the total cost of matrix-vector multiplication of our FS-1 algorithm for 2D Wasserstein-1 metric is reduced to $O(NM)$. The pseudo-code is presented in Algorithm 2.

This idea can be easily extended to high-dimensional cases, and we will not repeat it here. An argument similar to the one used in Section 3 shows that the FS-1 algorithm and the Sinkhorn algorithm in the 2D case still have the same stability. Thus, we shall omit the discussions.

5. Numerical experiments

In this section, we carry out four numerical experiments to evaluate the FS-1 algorithm. The first two examples show the performance of the FS-1 algorithm in 1D cases. Specifically, we consider the comparison of two 1D random distributions and the comparison of two Ricker wavelets arising from seismology [8]. The last two examples show the performance of the FS-1 algorithm in 2D cases. Specifically, we consider the comparison of two 2D random distributions and the comparison of two images arising from the image matching problem [39]. The marginal error $\|\text{diag}(\boldsymbol{\psi}) \times K^T \times \boldsymbol{\phi} - \boldsymbol{v}\|_1$ is chosen as the termination condition [3, 44]. All the experiments are conducted on a platform with 128G RAM, and one Intel(R) Xeon(R) Gold 5117 CPU @2.00GHz with 14 cores.

5.1. 1D random distributions.

We consider the Wasserstein-1 metric between two 1D random distributions on the interval $[-3,3]$. There are uniform grid points

$$x_i = (i-1)\Delta x - 3, \quad \Delta x = \frac{6}{N-1}, \quad i = 1, 2, \cdots, N.$$

Algorithm 2 2D FS-1 Algorithm

```
Input: $\boldsymbol{u},\boldsymbol{v}\in\mathbb{R}^{NM}$; itr_max $\in\mathbb{N}^+$; $h_1,h_2,\varepsilon,\mathrm{tol}\in\mathbb{R}$
Output: $W_\varepsilon(\boldsymbol{u},\boldsymbol{v})$
1: $\lambda_1\leftarrow e^{-h_1/\varepsilon};\lambda_2\leftarrow e^{-h_2/\varepsilon}$; $\boldsymbol{\phi},\boldsymbol{\psi}\leftarrow\frac{1}{NM}\mathbf{1}_{NM}$; $\boldsymbol{p},\boldsymbol{r},\boldsymbol{q},\boldsymbol{s}\leftarrow\mathbf{0}_{NM}$; $\ell\leftarrow 0$
2: while ($\ell<$ itr_max) and $\left(\sum_{i=1}^{M}\|\boldsymbol{\psi}_i\odot(\sum_{j=1}^{M}\lambda_2^{|i-j|}K_0\boldsymbol{\phi}_j)-\boldsymbol{v}_i\|_1>\mathrm{tol}\right)$ do
3:     $\boldsymbol{r}_1\leftarrow K_0\boldsymbol{\phi}_1$
4:     $\boldsymbol{s}_M\leftarrow\mathbf{0}_N$
5:     for $i=1:M-1$ do
6:         $\boldsymbol{r}_{i+1}\leftarrow\lambda_2\boldsymbol{r}_i+K_0\boldsymbol{\phi}_{i+1}$
7:         $\boldsymbol{s}_{M-i}\leftarrow\lambda_2(\boldsymbol{s}_{M-i+1}+K_0\boldsymbol{\phi}_{M-i+1})$
8:     $\boldsymbol{\psi}\leftarrow\boldsymbol{v}\oslash(\boldsymbol{r}+\boldsymbol{s})$
9:     $\boldsymbol{p}_1\leftarrow K_0\boldsymbol{\psi}_1$
10:    $\boldsymbol{q}_M\leftarrow\mathbf{0}_N$
11:    for $i=1:M-1$ do
12:        $\boldsymbol{p}_{i+1}\leftarrow\lambda_2\boldsymbol{p}_i+K_0\boldsymbol{\psi}_{i+1}$
13:        $\boldsymbol{q}_{M-i}\leftarrow\lambda_2(\boldsymbol{q}_{M-i+1}+K_0\boldsymbol{\psi}_{M-i+1})$
14:    $\boldsymbol{\phi}\leftarrow\boldsymbol{u}\oslash(\boldsymbol{p}+\boldsymbol{q})$
15:    $\ell\leftarrow\ell+1$
   return $\sum_{i_1,i_2=1}^{N}\sum_{j_1,j_2=1}^{M}\phi_{i_1,j_1}\psi_{i_2,j_2}\lambda_1^{|i_1-i_2|}\lambda_2^{|j_1-j_2|}(|i_1-i_2|h_1+|j_1-j_2|h_2)$
```

Correspondingly, we consider the two random vectors on the grid points

$$\boldsymbol{u}=(u_1,u_2,\cdots,u_N),\quad \boldsymbol{v}=(v_1,v_2,\cdots,v_N),$$

where u_i and v_j are both uniformly distributed on $[0,1]$. We would like to compare the performance and computational cost on computing the Wasserstein-1 metric $W_\varepsilon\left(\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|},\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}\right)$ using the Sinkhorn algorithm and the FS-1 algorithm. We tested 100 random experiments, and each experiment was performed for 1000 iterations.

In Table 5.1, we output the averaged computational time of two different algorithms. We can see that the FS-1 algorithm has an overwhelming advantage in computational speed. Moreover, the transport plans obtained by the two algorithms are almost identical. To further study the efficiency advantage of our FS-1 algorithm, we present the computational time of the two algorithms in different cases. By data fitting, we can see the empirical complexity of the FS-1 algorithm is $O(N^{1.02})$, while that of the Sinkhorn algorithm is $O(N^{2.22})$, see Figure 5.1 (Left) for illustration. In Figure 5.1 (Right), we discuss the computational time required to reach the corresponding marginal error under different regularization parameters ε for the random distribution with dimension $N=10000$. Obviously, the FS-1 algorithm is about two orders of magnitude faster than the Sinkhorn algorithm.

5.2. Ricker wavelet. Next, we consider the computation of the Wasserstein-1 metric between the Ricker wavelet

$$R(t)=A(1-2\pi^2f_0^2t^2)e^{-\pi^2f_0^2t^2},$$

and its translation $R(t-s)$. The Ricker wavelet is commonly used to model source time function in seismology [8]. Here f_0 is the dominant frequency, and A denotes the wave

N	Computational time (s)		Speed-up ratio	$\|P_{FS}-P\|_F$
	FS-1	Sinkhorn		
500	8.70×10^{-3}	7.68×10^{-2}	8.83×10^{0}	6.54×10^{-15}
2000	4.25×10^{-2}	2.81×10^{0}	6.61×10^{1}	4.98×10^{-18}
8000	1.53×10^{-1}	4.80×10^{1}	3.14×10^{2}	3.92×10^{-18}

TABLE 5.1. *The 1D random distribution problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points N. The regularization parameter $\varepsilon=0.001$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.*

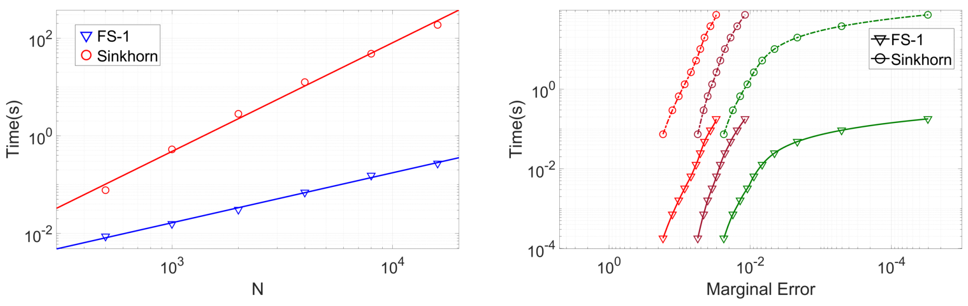

FIG. 5.1. *The 1D random distribution problem. Left: The comparison of computational time between the FS-1 algorithm and the Sinkhorn algorithm with different numbers of grid points N. Right: The computational time required to reach the corresponding marginal error under different regularization parameters $\varepsilon=0.1$ (Green), $\varepsilon=0.01$ (Purple), and $\varepsilon=0.001$ (Red).*

amplitude. For simplicity, we set

$$f_0=1,\quad A=1.$$

Since the Ricker wavelet is not always positive over the entire time duration, we will square and normalize it for the comparisons of the Wasserstein-1. In [27], a new normalization method with better convexity is given as follows:

$$D(f,g)=W_{1,\varepsilon}\left(\frac{\frac{f^2}{\|f^2\|}+\delta}{1+L\delta},\frac{\frac{g^2}{\|g^2\|}+\delta}{1+L\delta}\right), \tag{5.1}$$

where δ is a sufficiently small parameter to improve numerical stability, and L is a given parameter to guarantee that the two functions being compared are normalized.

Below, we randomly select the translation parameter $s=-1.2032$. This parameter can ensure that the two Ricker wavelets $R(t)$ and $R(t-s)$ are sufficiently far apart. And the small parameter $\delta=10^{-3}$. We repeated the experiment 100 times, and each experiment was performed for 500 iterations. In Table 5.2, we output the averaged computational time of two different algorithms. We also present the computational time required to reach the corresponding marginal error under different regularization parameters ε for the distribution with dimension $N=10000$, see Figure 5.2 (Left) for illustration, from which, we can draw the same conclusions as those in Subsection 5.1.

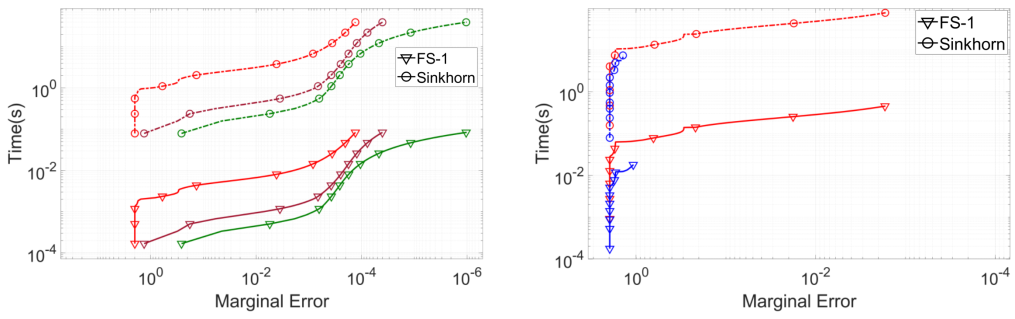

Fig. 5.2. *The Ricker wavelet problem. Left: The computational time required to reach the corresponding marginal error under different regularization parameters* $\varepsilon = 0.1$ *(Green),* $\varepsilon = 0.05$ *(Purple), and* $\varepsilon = 0.01$ *(Red). Right: The comparison between the Sinkhorn-type algorithms with (Red) and without (Blue) the log-domain stabilization for* $\varepsilon = 0.001$*.*

In Figure 5.2 (Right), we also discuss the impact of the log-domain stabilization technique for $\varepsilon = 0.001$. Without the technique, the Sinkhorn algorithm terminates abnormally at the 97th iteration and the FS-1 algorithm terminates abnormally at the 104th iteration. By introducing the log-domain stabilization technique, neither algorithm is terminated abnormally. Moreover, the stabilized FS-1 algorithm still maintains a significant efficiency advantage over the stabilized Sinkhorn algorithm.

N	Computational time (s)		Speed-up ratio	$\|\|P_{FS} - P\|\|_F$
	FS-1	Sinkhorn		
500	4.90×10^{-3}	2.30×10^{-2}	4.69×10^{0}	5.67×10^{-16}
2000	1.52×10^{-2}	1.36×10^{0}	8.93×10^{1}	1.81×10^{-17}
8000	5.96×10^{-2}	2.81×10^{1}	4.72×10^{2}	1.22×10^{-16}

Table 5.2. *The Ricker wavelet problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points* N*. The regularization parameter* $\varepsilon = 0.01$*. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.*

5.3. 2D Random distributions. Next, we discuss the performance of the FS-1 algorithm in two dimensions. This subsection generalizes the 1D random distributions in Subsection 5.1 to 2D random distributions. The basic settings are almost the same as in Subsection 5.1. And we need to compute the Wasserstein-1 metric between two $N \times N$ dimensional random vectors, where N is the number of grid points in each dimension. Without loss of generality, we set $h_1 = h_2 = 1$. We also tested 100 random experiments, and each experiment was performed for 1000 iterations. The averaged computational time for different numbers of nodes $N \times N$ of the two algorithms are output in Table 5.3 and Figure 5.3 (Left). By data fitting, we can see the empirical complexity of the FS-1 algorithm is $O(N^{1.66})$, while that of the Sinkhorn algorithm is $O(N^{4.44})$. These results are even better than the expected $O(N^2)$ complexity for the FS-1 algorithm. This again shows the big efficiency advantage of the FS-1 algorithm compared to the Sinkhorn

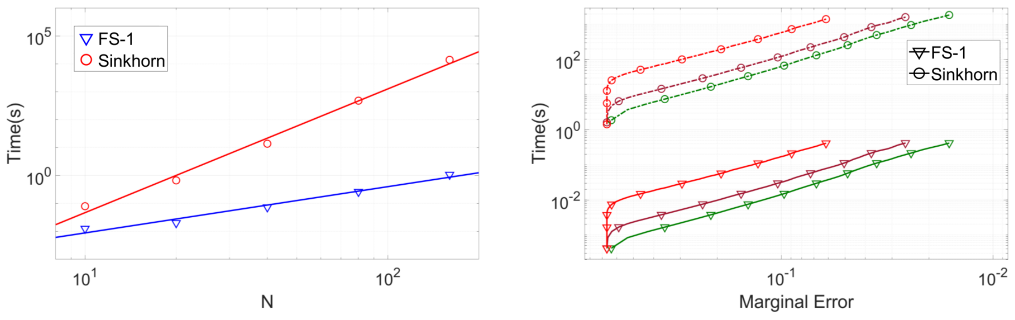

FIG. 5.3. *The 2D random distribution problem. Left: The comparison of computational time between the FS-1 algorithm and the Sinkhorn algorithm with different numbers of grid nodes N. Right: The computational time required to reach the corresponding marginal error under different regularization parameters $\varepsilon = 0.1$ (Green), $\varepsilon = 0.05$ (Purple), and $\varepsilon = 0.01$ (Red).*

algorithm. In Figure 5.3 (Right), we also present the computational time required to reach the corresponding marginal error under different regularization parameters ε for the random distribution with dimension 100×100. Obviously, the FS-1 algorithm still maintains the efficiency advantage for more than two orders of magnitude.

$N \times N$	Computational time (s)		Speed-up ratio	$\|\|P_{FS} - P\|\|_F$
	FS-1	Sinkhorn		
10×10	1.27×10^{-2}	8.02×10^{-2}	6.34×10^{0}	1.20×10^{-17}
20×20	2.01×10^{-2}	6.73×10^{-1}	3.34×10^{1}	5.96×10^{-18}
40×40	7.38×10^{-2}	1.38×10^{1}	1.87×10^{2}	3.00×10^{-18}
80×80	2.64×10^{-1}	4.78×10^{2}	1.81×10^{3}	1.55×10^{-18}
160×160	1.08×10^{0}	1.38×10^{4}	1.28×10^{4}	7.68×10^{-19}

TABLE 5.3. *The 2D random distribution problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different number of grid points $N \times N$. The regularization parameter $\varepsilon = 0.01$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.*

5.4. Image matching problem. An important application of the optimal transport in 2D is to match images. OT plays a fundamental role in related tasks including density regularization [6], image registration [18], and optical flow [10]. The complexity advantage of the FS-1 algorithm compared to the Sinkhorn algorithm can further enable practical applications of optimal transport in high-resolution images.

Here we consider the image matching experiment. We randomly select two images, see Figure 5.4 for illustration, from the DIV2K dataset [1], where the images have 2K pixels for at least one of the axes (vertical or horizontal). Considering that the resolutions of the images are different, we first sample them to the same scale $N \times N$. Without loss of generality, we set $h_1 = h_2 = 1$. In addition, to facilitate the optimal transport comparison, we convert them to grayscale images and normalize them using formula (5.1) with $\delta = 10^{-7}$. Again, we repeated the experiment 100 times and each experiment was performed for 1000 iterations.

FIG. 5.4. *The image matching problem. Illustration of images.*

In Table 5.4, we output the averaged computational time of two different algorithms. It should be emphasized that the computational time of the Sinkhorn algorithm is too long for large-scale images, so there is no result. However, the FS-1 algorithm can still handle this simply. We also present the computational time required to reach the corresponding marginal error under different regularization parameters ε. Here, the image size is 100×100, see Figure 5.5 (Left) for illustration, from which we can draw the same conclusions as those in Subsection 5.3.

In Figure 5.5 (Right), we also discuss the impact of the log-domain stabilization technique for $\varepsilon = 0.01$. Without the technique, the Sinkhorn algorithm and the FS-1 algorithm both terminate abnormally at the 138th iteration. By introducing the log-domain stabilization technique, neither algorithm is terminated abnormally. Moreover, the stabilized FS-1 algorithm still maintains a significant efficiency advantage over the stabilized Sinkhorn algorithm.

$N \times N$	Computational time (s)		Speed-up ratio	$\|P_{FS} - P\|_F$
	FS-1	Sinkhorn		
100×100	4.10×10^{-1}	1.32×10^{3}	3.23×10^{3}	2.28×10^{-17}
200×200	1.72×10^{0}	3.08×10^{4}	1.79×10^{4}	9.52×10^{-18}
400×400	7.23×10^{0}	–	–	–
800×800	3.20×10^{1}	–	–	–

TABLE 5.4. *The image matching problem. The comparison between the Sinkhorn algorithm and the FS-1 algorithm with the different total number of grid nodes $N \times N$. The regularization parameter $\varepsilon = 1$. Columns 2-4 are the averaged computational time of the two algorithms and the speed-up ratio of the FS-1 algorithm. Column 5 is the Frobenius norm of the difference between the transport plan computed by the two algorithms.*

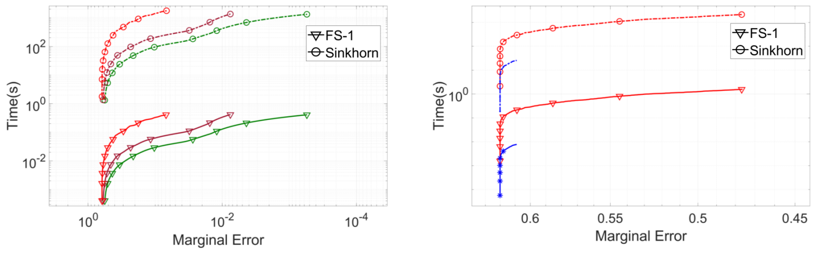

FIG. 5.5. *The image matching problem. Left: The computational time required to reach the corresponding marginal error under different regularization parameters $\varepsilon = 1$ (Green), $\varepsilon = 0.5$ (Purple), and $\varepsilon = 0.1$ (Red). Right: The comparison between the Sinkhorn-type algorithms with (Red) and without (Blue) the log-domain stabilization for $\varepsilon = 0.01$.*

6. Conclusion

In this paper, we propose an efficient (abbreviated as FS-1) algorithm to compute the Wasserstein-1 metric with linear computational cost per iteration. This method is developed by discovering the natural structure of Sinkhorn's kernel, which allows a matrix-vector multiplication to be carried out exactly with $O(N)$ cost for each iteration by using Qin Jiushao's or Horner's method for efficient polynomial evaluation. Moreover, the FS-1 algorithm can also be adapted to the widely used log-domain stabilization technique. As shown by numerous experiments, the FS-1 algorithm achieves a huge speed advantage without losing accuracy.

Finally, this paper mainly considers the acceleration of the Sinkhorn algorithm in the matrix-vector multiplication. It is well known that the number of iterations of the Sinkhorn algorithm will significantly increase with the increase of numerical accuracy, which leads to slow convergence. In [49], the Inexact Proximal point method for the Optimal Transport problem (IPOT) was proposed for this problem. We believe that FS-1 and IPOT can be effectively combined to develop a new algorithm for solving the Optimal Transport problem with fast convergence and low complexity. We are currently investigating this important extension and hope to report the progress in a future paper. Besides these, another interesting question is whether we can develop the FS-1 algorithm on the non-uniform mesh. There seems to be no direct answer, but it deserves further investigation.

Acknowledgements. This work was supported by National Natural Science Foundation of China Grant Nos.11871297 and 12031013, Tsinghua University Initiative Scientific Research Program, and Shanghai Municipal Science and Technology Major Project 2021SHZDZX0102.

REFERENCES

[1] E. Agustsson and R. Timofte, *NTIRE 2017 challenge on single image super-resolution: Dataset and study*, IEEE Conference on Computer Vision and Pattern Recognition Workshops, 1122-1131, 2017. 5.4

[2] J. Altschuler, F. Bach, A. Rudi, and J. Weed, *Massively scalable Sinkhorn distances via the Nyström method*, Adv. Neural Inf. Process. Syst., 32:4427–4437, 2019. 1

[3] J. Altschuler, J. Weed, and P. Rigollet, *Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration*, Adv. Neural Inf. Process. Syst., 30:1964–1974, 2017. 5

[4] J.D. Benamou and Y. Brenier, *A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem*, Numer. Math., 84:375–393, 2000. 1

[5] 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., 260:107–126, 2014. 1

[6] M. Burger, M. Franek, and C.B. Schönlieb, *Regularized regression and density estimation based on optimal transport*, Appl. Math. Res. Express, 2012:209–253, 2012. 5.4

[7] G. Buttazzo, L. De Pascale, and P. Gori-Giorgi, *Optimal-transport formulation of electronic density-functional theory*, Phys. Rev. A, 85:062502, 2012. 1

[8] J. Chen, Y. Chen, H. Wu, and D. Yang, *The quadratic Wasserstein metric for earthquake location*, J. Comput. Phys., 373:188–209, 2018. 1, 5, 5.2

[9] L. Chizat, G. Peyré, B. Schmitzer, and F.-X. Vialard, *Scaling algorithms for unbalanced optimal transport problems*, Math. Comput., 87:2563–2609, 2018. 1, 2.1, 3, 3.2

[10] P. Clarysse, B. Delhay, M. Picq, and J. Pousin, *Optimal extended optical flow subject to a statistical constraint*, J. Comput. Appl. Math., 234:1291–1302, 2010. 5.4

[11] P.L. Combettes and J.C. Pesquet, *Proximal splitting methods in signal processing*, in H. Bauschke, R. Burachik, P. Combettes, V. Elser, D. Luke, and H. Wolkowicz (eds.), Fixed-Point Algorithms for Inverse Problems in Science and Engineering, Springer, 185–212, 2011. 1

[12] C. Cotar, G. Friesecke, and C. Klüppelberg, *Density functional theory and optimal transportation with Coulomb cost*, Commun. Pure Appl. Math., 66:548–599, 2013. 1

[13] M. Cuturi, *Sinkhorn distances: Lightspeed computation of optimal transport*, Adv. Neural Inf. Process. Syst., 26:2292–2300, 2013. 1, 2

[14] B. Engquist, K. Ren, and Y. Yang, *The quadratic Wasserstein metric for inverse data matching*, Inverse Probl., 36:055001, 2020. 1

[15] B.D. Froese, *Numerical methods for the elliptic Monge-Ampere equation and optimal transport*, PhD thesis, Simon Fraser University, 2012. 1

[16] B.D. Froese and A.M. Oberman, *Convergent finite difference solvers for viscosity solutions of the elliptic Monge–Ampère equation in dimensions two and higher*, SIAM J. Numer. Anal., 49:1692–1714, 2011. 1

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Generative adversarial nets*, Adv. Neural Inf. Process. Syst., 27:2672–2680, 2014. 1

[18] S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent, *Optimal mass transport for registration and warping*, Int. J. Comput. Vis., 60:225–240, 2004. 5.4

[19] H. Heaton, S.W. Fung, A.T. Lin, S. Osher, and W. Yin, *Wasserstein-based projections with applications to inverse problems*, SIAM J. Math. Data Sci., 4(2):581–603, 2022 1

[20] W.G. Horner, *XXI. A new method of solving numerical equations of all orders, by continuous approximation*, Philos. Trans. R. Soc. Lond., 109:308–335, 1819. 1, 3.1

[21] Y. Hu, H. Chen, and X. Liu, *A global optimization approach for multi-marginal optimal transport problems with Coulomb cost*, arXiv preprint, arXiv:2110.07352, 2021. 1

[22] J. Kleinberg and E. Tardos, *Algorithm Design*, Pearson Education India, 2006. 1

[23] J. Klicpera, M. Lienen, and S. Günnemann, *Scalable optimal transport in high dimensions for graph distances, embedding alignment, and more*, Proc. Mach. Learn. Res., 139:5616–5627, 2021. 1

[24] T. Le, M. Yamada, K. Fukumizu, and M. Cuturi, *Tree-sliced variants of Wasserstein distances*, Adv. Neural Inf. Process. Syst., 32:12304–12315, 2019. 1

[25] W. Li, E.K. Ryu, S. Osher, W. Yin, and W. Gangbo, *A parallel method for earth mover distance*, J. Sci. Comput., 75:182–197, 2018. 1

[26] X. Li, D. Sun, and K.C. Toh, *An asymptotically superlinearly convergent semismooth Newton augmented Lagrangian method for linear programming*, SIAM J. Optim., 30:2410–2440, 2020. 1

[27] Z. Li, Y. Tang, J. Chen, and H. Wu, *The quadratic Wasserstein metric with squaring scaling for seismic velocity inversion*, arXiv preprint, arXiv:2201.11305, 2022. 5.2

[28] A.T. Lin, W. Li, S. Osher, and G. Montúfar, *Wasserstein proximal of GANs*, in F. Nielsen and F. Barbaresco (eds.), Geometric Science of Information. GSI 2021. Lecture Notes in Computer Science, Springer, 524–533, 2021. 1

[29] T. Lin, N. Ho, and M.I. Jordan, *On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms*, Proc. Mach. Learn. Res., 97:3982-3991, 2019. 1

[30] J. Liu, W. Yin, W. Li, and Y.T. Chow, *Multilevel optimal transport: A fast approximation of Wasserstein-1 distances*, SIAM J. Sci. Comput., 43:A193–A220, 2021. 1

[31] C. Meng, Y. Ke, J. Zhang, M. Zhang, W. Zhong, and P. Ma, *Large-scale optimal transport map estimation using projection pursuit*, Adv. Neural Inf. Process. Syst., 32:8118–8129, 2019. 1

[32] C. Meng, J. Yu, J. Zhang, P. Ma, and W. Zhong, *Sufficient dimension reduction for classification using principal optimal transport direction*, Adv. Neural Inf. Process. Syst., 33:4015–4028, 2020. 1

[33] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux, *Measuring the misfit between seismograms using an optimal transport distance: Application to full waveform inversion*, Geophys. J. Int., 205:345–377, 2016. 1

[34] L. Métivier, R. Brossier, Q. Merigot, É. Oudet, and J. Virieux, *An optimal transport approach for seismic tomography: Application to 3D full waveform inversion*, Inverse Probl., 32:115008, 2016. 1

[35] J. Needham, *Science and Civilisation in China*, Cambridge University Press, 5, 1974. 1, 3.1

[36] O. Pele and M. Werman, *Fast and robust earth mover's distances*, in 2009 IEEE International Conference on Computer Vision, 460–467, 2009. 1

[37] G. Peyré and M. Cuturi, *Computational optimal transport: With applications to data science*, Found. Trends Mach. Learn., 11:355–607, 2019. 1

[38] Y. Rubner, C. Tomasi, and L.J. Guibas, *The earth mover's distance as a metric for image retrieval*, Int. J. Comput. Vis., 40:99–121, 2000. 1

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, and L. Fei-Fei, *ImageNet large scale visual recognition challenge*, Int. J. Comput. Vis., 115:211–252, 2015. 5

[40] E.K. Ryu, Y. Chen, W. Li, and S. Osher, *Vector and matrix optimal mass transport: theory, algorithm, and applications*, SIAM J. Sci. Comput., 40:A3675–A3698, 2018. 1

[41] 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., 75:1596–1613, 2018. 1

[42] F. Santambrogio, *Optimal Transport for Applied Mathematicians*, Birkäuser, NY, 55:94, 2015. 1

[43] M. Scetbon and M. Cuturi, *Linear time Sinkhorn divergences using positive features*, Adv. Neural Inf. Process. Syst., 33:13468–13480, 2020. 1

[44] M. Scetbon, M. Cuturi, and G. Peyré, *Low-rank Sinkhorn factorization*, Proc. Mach. Learn. Res., 139:9344-9354, 2021. 5

[45] R. Sinkhorn, *Diagonal equivalence to matrices with prescribed row and column sums*, Am. Math. Mon., 74:402–405, 1967. 1, 2, 2

[46] J. Solomon, F. De Goes, G. Peyré, M. Cuturi, A. Butscher, A. Nguyen, T. Du, and L. Guibas, *Convolutional Wasserstein distances: Efficient optimal transportation on geometric domains*, ACM Trans. Graph., 34:1–11, 2015. 1

[47] Y. Takezawa, R. Sato, and M. Yamada, *Supervised tree-Wasserstein distance*, Proc. Mach. Learn. Res., 139:10086-10095, 2021. 1

[48] C. Villani, *Optimal Transport: Old and New*, Springer, 338, 2009. 2

[49] Y. Xie, X. Wang, R. Wang, and H. Zha, *A fast proximal point method for computing exact Wasserstein distance*, Proc. Mach. Learn. Res., 115:433-453, 2020. 6

[50] L. Yang, J. Li, D. Sun, and K.-C. Toh, *A fast globally linearly convergent algorithm for the computation of Wasserstein barycenters*, J. Mach. Learn. Res., 22:1–37, 2021. 1

[51] Y. Yang, B. Engquist, J. Sun, and B.F. Hamfeldt, *Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion*, Geophysics, 83:R43–R62, 2018. 1